HANDBOOK OF GEOPHYSICAL EXPLORATION
SEISMIC EXPLORATION

Klaus Helbig *and* Sven Treitel, Editors

VOLUME 30

Computational Neural Networks for Geophysical Data Processing

edited by M.M. POULTON

PERGAMON

HANDBOOK OF GEOPHYSICAL EXPLORATION

SEISMIC EXPLORATION

Editors: Klaus Helbig and Sven Treitel

Volume

1. Basic Theory in Reflection Seismology[1]
2. Seismic Instrumentation, 2nd Edition[1]
3. Seismic Field Techniques[2]
4A. Seismic Inversion and Deconvolution: Classical Methods
4B. Seismic Inversion and Deconvolution: Dual-Sensor Technology
5. Seismic Migration (Theory and Practice)
6. Seismic Velocity Analysis[1]
7. Seismic Noise Attenuation
8. Structural Interpretation[2]
9. Seismic Stratigraphy
10. Production Seismology
11. 3-D Seismic Exploration[2]
12. Seismic Resolution
13. Refraction Seismics
14. Vertical Seismic Profiling: Principles
3rd Updated and Revised Edition
15A. Seismic Shear Waves: Theory
15B. Seismic Shear Waves: Applications
16A. Seismic Coal Exploration: Surface Methods[2]
16B. Seismic Coal Exploration: In-Seam Seismics
17. Mathematical Aspects of Seismology
18. Physical Properties of Rocks
19. Shallow High-Resolution Reflection Seismics
20. Pattern Recognition and Image Processing
21. Supercomputers in Seismic Exploration
22. Foundations of Anisotropy for Exploration Seismics
23. Seismic Tomography[2]
24. Borehole Acoustics[1]
25. High Frequency Crosswell Seismic Profiling[2]
26. Applications of Anisotropy in Vertical Seismic Profiling[1]
27. Seismic Multiple Elimination Techniques[1]
28. Wavelet Transforms and Their Applications to Seismic Data
Acquisition, Compression, Processing and Interpretation[1]
29. Seismic Signatures and Analysis of Reflection Data
in Anisotropic Media
30. Computational Neural Networks for Geophysical Data Processing

[1] In preparation.

[2] Planned.

This Page Intentionally Left Blank

SEISMIC EXPLORATION

Volume 30

COMPUTATIONAL NEURAL NETWORKS FOR GEOPHYSICAL DATA PROCESSING

edited by

Mary M. POULTON
Department of Mining & Geological Engineering
Computational Intelligence & Visualization Lab.
The University of Arizona
Tucson, AZ 85721-0012
USA

2001
PERGAMON
An Imprint of Elsevier Science
Amsterdam – London – New York – Oxford – Paris – Shannon – Tokyo

ELSEVIER SCIENCE Ltd
The Boulevard, Langford Lane
Kidlington, Oxford OX5 1GB, UK

© 2001 Elsevier Science Ltd. All rights reserved.

This work is protected under copyright by Elsevier Science, and the following terms and conditions apply to its use:

Photocopying
Single photocopies of single chapters may be made for personal use as allowed by national copyright laws. Permission of the Publisher and payment of a fee is required for all other photocopying, including multiple or systematic copying, copying for advertising or promotional purposes, resale, and all forms of document delivery. Special rates are available for educational institutions that wish to make photocopies for non-profit educational classroom use.

Permissions may be sought directly from Elsevier Science Global Rights Department, PO Box 800, Oxford OX5 1DX, UK; phone: (+44) 1865 843830, fax: (+44) 1865 853333, e-mail: permissions@elsevier.co.uk. You may also contact Global Rights directly through Elsevier's home page (http://www.elsevier.nl), by selecting 'Obtaining Permissions'.

In the USA, users may clear permissions and make payments through the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923, USA; phone: (+1) (978) 7508400, fax: (+1) (978) 7504744, and in the UK through the Copyright Licensing Agency Rapid Clearance Service (CLARCS), 90 Tottenham Court Road, London W1P 0LP, UK; phone: (+44) 207 631 5555; fax: (+44) 207 631 5500. Other countries may have a local reprographic rights agency for payments.

Derivative Works
Tables of contents may be reproduced for internal circulation, but permission of Elsevier Science is required for external resale or distribution of such material.
Permission of the Publisher is required for all other derivative works, including compilations and translations.

Electronic Storage or Usage
Permission of the Publisher is required to store or use electronically any material contained in this work, including any chapter or part of a chapter.

Except as outlined above, no part of this work may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without prior written permission of the Publisher.
Address permissions requests to: Elsevier Science Global Rights Department, at the mail, fax and e-mail addresses noted above.

Notice
No responsibility is assumed by the Publisher for any injury and/or damage to persons or property as a matter of products liability, negligence or otherwise, or from any use or operation of any methods, products, instructions or ideas contained in the material herein. Because of rapid advances in the medical sciences, in particular, independent verification of diagnoses and drug dosages should be made.

First edition 2001

Library of Congress Cataloging in Publication Data
A catalog record from the Library of Congress has been applied for.

British Library Cataloguing in Publication Data
A catalogue record from the British Library has been applied for.

ISBN: 0-08-043986-1
ISSN: 0950-1401 (Series)

♾ The paper used in this publication meets the requirements of ANSI/NISO Z39.48-1992 (Permanence of Paper).
Printed in The Netherlands.

TABLE OF CONTENTS

Preface xi
Contributing Authors xiii

Part I Introduction to Computational Neural Networks 1
Chapter 1 A Brief History 3
1. Introduction 3
2. Historical Development 5
2.1. Mcculloch and Pitts Neuron 7
2.2. Hebbian Learning 9
2.3. Neurocomputing 10
2.4. Perceptron 11
2.5. ADALINE 13
2.6. Caianiello Neurons 14
2.7. Limitations 15
2.8. Next Generation 15

Chapter 2 Biological Versus Computational Neural Networks 19
1. Computational Neural Networks 19
2. Biological Neural Networks 19
3. Evolution of the Computational Neural Network 23

Chapter 3 Multi-Layer Perceptrons and Back-Propagation Learning 27
1. Vocabulary 27
2. Back-Propagation 28
3. Parameters 35
3.1. Number of Hidden Layers 35
3.2. Number of Hidden Pes 37
3.3. Threshold Function 43
3.4. Weight Initialization 43
3.5. Learning Rate and Momentum 45
3.6. Bias 46
3.7. Error Accumulation 47
3.8. Error Calculation 49
3.9. Regularization and Weight Decay 49
4. Time-Varying Data 50

Chapter 4 Design of Training and Testing Sets 55
1. Introduction 55
2. Re-Scaling 56

3. Data Distribution 58
4. Size Reduction 58
5. Data Coding 60
6. Order of Data 61

Chapter 5 Alternative Architectures and Learning Rules 66
1. Improving on Back-Propagation 66
  1.1. Delta Bar Delta 67
  1.2. Directed Random Search 68
  1.3. Resilient Back-Propagation 69
  1.4. Conjugate Gradient 71
  1.5. Quasi-Newton Method 72
  1.6. Levenberg-Marquardt 72
2. Hybrid Networks 74
  2.1. Radial Basis Function Network 74
  2.2. Modular Neural Network 75
  2.3. Probabilistic Neural Network 75
  2.4. Generalized Regression Neural Network 78
3. Alternative Architectures 78
  3.1. Self Organizing Map 78
  3.2. Hopfield Networks 81
  3.3. Adaptive Resonance theory 84

Chapter 6 Software and Other Resources 89
1. Introduction 89
2. Commercial Software Packages 89
3. Open Source Software 97
4. News Groups 97

Part II Seismic Data Processing 99
Chapter 7 Seismic Interpretation and Processing Applications 101
1. Introduction 101
2. Waveform Recognition 101
3. Picking Arrival Times 103
4. Trace Editing 109
5. Velocity Analysis 110
6. Elimination of Multiples 112
7. Deconvolution 113
8. Inversion 115

Chapter 8 Rock Mass and Reservoir Characterization 119
1. Introduction 119
2. Horizon Tracking and Facies Maps 119

3. Time-Lapse Interpretation 121
4. Predicting Log Properties 121
5. Rock/Reservoir Characterization 124

Chapter 9 Identifying Seismic Crew Noise 129
1. Introduction 129
  1.1. Current Attenuation Methods 129
  1.2. Patterns of Crew Noise Interference 131
  1.3. Pre-Processing 134
2. Training Set Design and Network Architecture 134
  2.1. Selection of Interference Training Examples 135
  2.2. Selection of Signal Training Patterns 138
3. Testing 139
4. Analysis of Training and Testing 141
  4.1. Sensitivity to Class Distribution 142
  4.2. Sensitivity to Network Architecture 144
  4.3. Effect of Confidence Level During Overlapping Window Tabulation 147
  4.4. Effect of NMO Correction 148
5. Validation 150
  5.1. Effect on Deconvolution 150
  5.2. Effect on CMP Stacking 151
6. Conclusions 153

Chapter 10 Self-Organizing Map (SOM) Network for Tracking Horizons and Classifying Seismic Traces 155
1. Introduction 155
2. Self-Organizing Map Network 155
3. Horizon Tracking 157
  3.1. Training Set 157
  3.2. Results 158
4. Classification of the Seismic Traces 161
  4.1. Window Length and Placement 166
  4.2. Number of Classes 168
5. Conclusions 169

Chapter 11 Permeability Estimation with an RBF Network and Levenberg-Marquardt Learning 171
1. Introduction 171
2. Relationship Between Seismic and Petrophysical Parameters 172
  2.1. RBF Network Training 173
  2.2. Predicting Hydraulic Properties From Seismic Information: Relation Between Velocity and Permeability 174
3. Parameters That Affect Permeability: Porosity, Grain Size, Clay Content 176

4. Neural Network Modeling of Permeability Data 178
  4.1. Data Analysis and Interpretation 181
  4.2. Assessing the Relative Importance of Individual Input Attributes 182
5. Summary and Conclusions 184

Chapter 12 Caianiello Neural Network Method for Geophysical Inverse Problems 187
1. Introduction 187
2. Generalized Geophysical Inversion 188
  2.1. Generalized Geophysical Model 188
  2.2. Ill-Posedness and Singularity 190
  2.3. Statistical Strategy 192
  2.4. Ambiguous Physical Relationship 193
3. Caianiello Neural Network Method 194
  3.1. Mcculloch-Pitts Neuron Model 194
  3.2. Caianiello Neuron Model 194
  3.3. The Caianiello Neuron-Based Multi-Layer Network 195
  3.4. Neural Wavelet Estimation 196
  3.5. Input Signal Reconstruction 198
  3.6. Nonlinear Factor Optimization 198
4. Inversion With Simplified Physical Models 199
  4.1. Simplified Physical Model 199
  4.2. Joint Impedance Inversion Method 200
  4.3. Nonlinear Transform 201
  4.4. Joint Inversion Step 1: MSI and MS Wavelet Extraction At the Wells 202
  4.5. Joint Inversion Step 2: Initial Impedance Model Estimation 204
  4.6. Joint Inversion Step 3: Model-Based Impedance Improvement 204
  4.7. Large-Scale Stratigraphic Constraint 205
5. Inversion With Empirically-Derived Models 206
  5.1. Empirically Derived Petrophysical Model for the Trend 206
  5.2. Neural Wavelets for Scatter Distribution 207
  5.3. Joint Inversion Strategy 207
6. Example 208
7. Discussions and Conclusions 210

Part III Non-Seismic Applications 217
Chapter 13 Non-Seismic Applications 219
1. Introduction 219
2. Well Logging 220
  2.1. Porosity and Permeability Estimation 220
  2.2. Lithofacies Mapping 221
3. Gravity and Magnetics 224
4. Electromagnetics 225

  - 4.1. Frequency-Domain 225
  - 4.2. Time-Domain 227
  - 4.3. Magnetotelluric 227
  - 4.4.. Ground Penetrating Radar 229
- 5. Resistivity 229
- 6. Multi-Sensor Data 230

Chapter 14 Detection of AEM Anomalies Corresponding to Dike Structures 234
- 1. Introduction 234
- 2. Airborne Electromagnetic Method - Theoretical Background 236
  - 2.1. General 236
  - 2.2 Forward Modeling for 1 Dimensional Models 237
  - 2.3. Forward Modelling for 2 Dimensional Models With EMIGMA 239
- 3. Feedforward Computational Neural Networks (CNN) 240
- 4. Concept 243
- 5. CNNs to Calculate Homogeneous Halfspaces 244
- 6. CNN for Detecting 2D Structures 247
  - 6.1. Training and Test Vectors 247
  - 6.2. Calculation of the Error Term (±1ppm, ±2ppm) 249
  - 6.3. Calculation of the Random Models (Model Categories 6-8) 249
  - 6.4. Training 249
- 7. Testing 250
- 8. Conclusion 252

Chapter 15 Locating Layer Boundaries with Unfocused Resistivity Tools 257
- 1. Introduction 257
- 2. Layer Boundary Picking 260
- 3. Modular Neural Network 262
- 4. Training With Multiple Logging Tools 265
  - 4.1. Mnn, Mlp, and Rbf Architectures 266
  - 4.2. Rprop and Grnn Architectures 267
- 5. Analysis of Results 268
  - 5.1. Thin Layer Model (Thickness From 0.5 to 2 M) 268
  - 5.2. Medium-Thickness Layer Model (Thickness From 1.5 to 4 M) 273
  - 5.3. Thick Layer Model (Thickness From 6 to 16 M) 277
  - 5.4. Testing the Sensitivity to Resistivity 280
- 6. Conclusions 283

Chapter 16 A Neural Network Interpretation System for Near-Surface Geophysics Electromagnetic Ellipticity Soundings 286
- 1. Introduction 286

2. Function Approximation 289
  2.1. Background 289
  2.2. Radial Basis Function Neural Network 290
3. Neural Network Training 294
4. Case History 297
  4.1. Piecewise Half-Space Interpretation 298
  4.2. Half-Space Interpretations 302
5. Conclusion 303

Chapter 17 Extracting IP Parameters From TEM Data 307
1. Introduction 307
2. Forward Modeling 310
3. Inverse Modeling With Neural Networks 310
4. Testing Results 311
  4.1. Half-Space 311
  4.2. Layered Ground 312
  4.3. Polarizable First Layer 312
  4.4. Polarizable Second Layer 316
5. Uncertainty Evaluation 320
6. Sensitivity Evaluation 321
7. Case Study 321
8. Conclusions 324

Author Index 327
Index 331

PREFACE

I have been working in the field of neural network computing for the past 14 years, primarily in applied geophysics. During that time I have had the opportunity to train many graduate and undergraduate students and colleagues in the use of neural networks. At some point during their training, or during the course I teach on applied neural network computing there is always an "aha" moment when the vocabulary and concepts come together and the students have grasped the fundamental material and are ready to learn about the details. My goal in writing this book is to present the subject to an audience that has heard about neural networks or has had some experience with the algorithms but has not yet had that "aha" moment. For those that already have a solid grasp of how to create a neural network application, the book can provide a wide range of examples of nuances in network design, data set design, testing strategy, and error analysis.

There are many excellent books on neural networks and all are written from a particular perspective, usually signal processing, process control, or image processing. None of the books capture the full range of applications in applied geophysics or present examples relevant to problems of interest in geophysics today. Much of the success of a neural network application depends on a solid understanding of the data and a solid understanding of how to construct an appropriate data set, network architecture, and validation strategy. While this book cannot provide a blue print for every conceivable geophysics application, it does outline a basic approach that I have used successfully on my projects.

I use computational, rather than artificial, as the modifier for neural networks in this book to make a distinction between networks that are implemented in hardware and those that are implemented in software. The term artificial neural network covers any implementation that is inorganic and is the most general term. Computational neural networks are only implemented in software but represent the vast majority of applications.

The book is divided into three major sections: Introductory Theory (Chapters 1-6); Seismic Applications (Chapters 7-12); and Non-Seismic Applications (Chapters 13-17). Chapters contributed by other authors were selected to illustrate particular aspects of network design or data issues along with specific applications. Chapters 7 and 8 present a survey of the literature in seismic applications with emphasis on oil and gas exploration and production. Chapter 9 deals with crew noise in marine surveys and emphasizes how important training set design is to the success of the application. Chapter 10 illustrates one of the most popular applications of neural networks in the oil and gas industry – the use of an architecture that finds similarities between seismic wavelets with very little user interaction. Chapter 11 compares a neural network approach with regression. Chapter 12 is included to outline a seismic inversion approach with neural networks.

In the Non-Seismic section, Chapter 13 discusses applications in well logging, potential fields, and electrical methods. Chapter 14 introduces alternative cost functions in the context of an airborne electromagnetic survey. Chapter 15 compares several different architectures and learning strategies for a well logging interpretation problem. Chapter 16 compares neural network estimates to more conventional least-squares inversion results for a frequency-domain electromagnetic survey. Chapter 17 presents a method to attach a confidence measure to neural network-estimated model parameters in a time-domain electromagnetic survey. Each chapter introduces a different architecture or learning algorithm.

The notation used in the book presents vectors with a superscript arrow. In Chapter 12, however, vectors are denoted with bold letters for the sake of readability in the equations. Matrices are capital letters. Subscripts generally denote individual processing elements in a network with i indicating the input layer, j the hidden layer, k the output layer, and p and individual pattern.

I would like to thank all those who helped me while the book was in progress. My husband William and son Alexander decided that they had lived with the book for so long they may as well name it and adopt it. My editor at Elsevier, Friso Veneestra, patiently waited for the book to materialize and helped in the final stages of preparation. My copy editor Dorothy Peltz gave up retirement and worked long days to find mistakes and inconsistencies in the chapters. The layout editor Wendy Stewart learned more about the idiosyncrasies of Word than any sane human should know. John Greenhouse, James Fink, Wayne Pennington, Anna and Ferenc Szidarovszky spent countless hours reviewing the manuscript and providing valuable comments to improve the book. The students in my applied neural network computing course agreed to be guinea pigs and use the first six chapters as their textbook and provided valuable input. Thank you Chris, Michael, Bill, Kathy, David, Louis, Lewis, Mofya, Deny, Randy, Prachi, and Anna. And finally, I want to thank all my graduate students in the Computational Intelligence and Visualization Laboratory, past and present, who have shared my enthusiasm for the subject and contributed to the book.

Mary M. Poulton
Tucson, Arizona

CONTRIBUTING AUTHORS

Andreas Ahl University of Vienna	Chapter 14 Detection of AEM Anomalies Corresponding to Dike Structures
Ralf A. Birken Witten Technologies	Chapter 16 A Neural Network Interpretation System for Near-Surface Geophysics Electromagnetic Ellipticity Soundings
Fred K. Boadu Duke University	Chapter 11 Permeability Estimation with an RBF Network and Levenberg-Marquardt Learning
Vinton B. Buffenmyer ExxonMobil	Chapter 9 Identifying Seismic Crew Noise
Hesham El-Kaliouby The University of Arizona	Chapter 17 Extracting IP Parameters from TEM Data
Li-Yun Fu CSIRO Petroleum	Chapter 12 Caianiello Neural Network Method for Geophysical Inverse Problems
Meghan S. Miller USGS Menlo Park	Chapter 7 Seismic Interpretation and Processing Applications
Kathy S. Powell The University of Arizona	Chapter 7 Seismic Interpretation and Processing Applications Chapter 8 Rock Mass and Reservoir Characterization
John Quieren Halliburton	Chapter 10 Self-Organizing Map (SOM) Network for Tracking Horizons and Classifying Seismic Traces
James S. Schuelke ExxonMobil	Chapter 10 Self-Organizing Map (SOM) Network for Tracking Horizons and Classifying Seismic Traces
Wolfgang Seiberl University of Vienna	Chapter 14 Detection of AEM Anomalies Corresponding to Dike Structures
Lin Zhang Chevron	Chapter 10 Self-Organizing Map (SOM) Network for Tracking Horizons and Classifying Seismic Traces Chapter 15 Locating Layer Boundaries with Unfocused Resistivity Tools

This Page Intentionally Left Blank

Part I
Introduction to Computational Neural Networks

The first six chapters of this book provide a history of computational neural networks, a brief background on their biological roots, and an overview of the architectures, learning algorithms, and training requirements. Chapter 6 provides a review of major software packages and commercial freeware.

Computational neural networks are not faithful anatomical models of biological nervous systems but they can be considered physiological models. In other words, they do not attempt to mimic neural activity at a chemical or molecular level but they can model the function of biological networks, albeit at a very simple level. Computational neural network models are usually based on the cerebral cortex. The cerebrum is the largest structure in the brain. The convoluted outer surface of the cerebrum is the cortex, which performs the functions that allow us to interact with our world, make decisions, judge information, and make associations. The cerebrum first appeared in our ancestors nearly 200 million years ago[1]. Hence the types of functions the networks of neurons in the cortex have grown to excel at are those functions that provided an advantage for survival and growth – making sense of a complicated environment through pattern recognition, association, memory, organization of information, and understanding.

Computational neural networks are designed to automate complex pattern recognition tasks. Because the computational neural networks are mathematical tools, they can quantify patterns and estimate parameters. When computational neural networks became widely applied in the late 1980s, their performance was usually compared to statistical classification methods and regression. The conclusion from hundreds of comparisons on a variety of problems was that, in the worst case, the neural networks performed as well as the traditional methods of classification or function estimation and in most cases they performed significantly better. The application chapters in Parts II and III of this book will make few comparisons to other techniques because when properly applied the networks will perform at least as well as any other method and often better.

The focus in this book will be on processing static rather than time-varying data. But Chapters 1 and 3 have a brief description of dealing with time-varying data and Chapter 12 develops a network model specifically for time sequences.

Neural networks are no harder to use than statistical methods. Many of the issues surrounding construction of training and testing sets for a neural network are identical to the data needs of other techniques. Part I of this book should provide the reader with enough background to begin to work with neural networks and better understand the existing and potential applications to geophysics.

---

[1] Ornstein, R., and Thompson, R., 1984, The Amazing Brain: Houghton-Mifflin.

This Page Intentionally Left Blank

Chapter 1
A Brief History

Mary M. Poulton

1. INTRODUCTION

Computational neural networks are not just the grist of science fiction writers anymore nor are they a flash in the pan that will soon fade from use. The field of computational neural networks has matured in the last decade and found so many industrial applications that the notion of using a neural network to solve a particular problem no longer needs a "sales pitch" to management in many companies. Neural networks are now being routinely used in process control, manufacturing, quality control, product design, financial analysis, fraud detection, loan approval, voice and handwriting recognition, and data mining to name just a few application areas. The anti-virus software on your computer probably uses neural networks to recognize bit patterns related to viruses. When you buy a product on the Internet a neural network may be analyzing your buying patterns and predicting what products should be advertised to you.

Interest in computational intelligence techniques in the geophysical community has also increased in the last decade. The graph in Figure 1.1 shows the number of neural network papers with geophysical applications published in each of the last 10 years. One indicator of the maturity of neural network research in a discipline is the number of different journals and conference proceedings in which such papers appear. Figure 1.2 shows the number of different geophysical journals and conferences publishing neural network papers in the past 10 years. The numbers of papers shown in the figures are approximate since even the major bibliographic databases do not capture all the conferences, journals, or student theses in geophysics. While the number of papers published in 1998 may not be complete, the trend for the mid- to late-1990s seems to suggest that the field has matured beyond exploring all the possible applications of neural networks to geophysical data processing and is now focused on the most promising applications.

Biological neural networks are "trained" from birth to be associative memories. An input stimulus becomes associated with an object, a behavior, a sensation, etc. Dense networks of neurons in the brain retain some memory of the association between input patterns received from our external sensory organs and the meaning of those patterns. Over time we are able to look at many variations of the same pattern and associate them all with the same class of object. For example, infants see many examples of human faces, animal faces, and cartoon faces and eventually learn to associate certain key characteristics of each with the appropriate class designation. We could program a computer to perform the same task using mathematical equations that specifically describe each face. Or, we could encode each face as a digital image, present the image to a computational neural network along with a class label of "human", "cartoon", or "animal" and let the network make the associations between the images and the labels without receiving any more explicit descriptions of the faces from us. The associations we ask a computational neural network to make can take the form of a classification described above or a regression where we want to estimate a particular value based on an input pattern. In either case, the computational neural network performs as a function approximator and the field draws upon the large body of research from estimation theory, inverse theory, Bayesian statistics, and optimization theory.

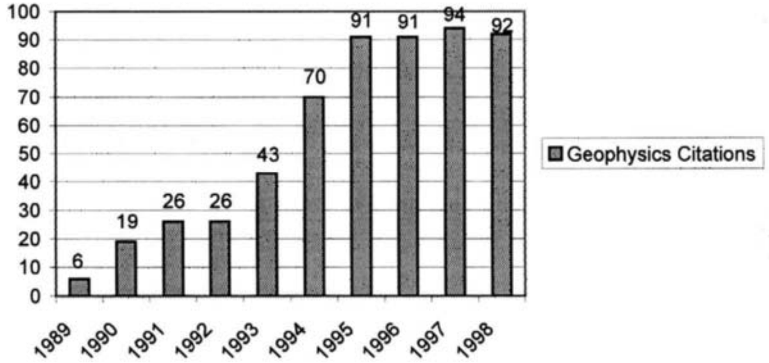


Figure 1.1. Numbers of journal articles, conference papers, reports and theses on application of neural networks to geophysics. Papers related to applied geophysics topics such as petroleum and mineral exploration and environmental engineering are shown separately.

When we analyze geophysical data we are looking for patterns associated with particular "targets". Those targets are either geological in nature such as a gas or oil horizon, an aquifer, or a mineral deposit; or the targets are human-made but have an interaction with the earth such as hazardous waste, unexploded ordnance, tunnels, etc. In either case we can measure a physical response attributed to the target that is different to the response from the earth if the target was not present. As geophysicists, we learn to associate the target response with the class or characteristics of a target. Computational neural networks can also learn to make those associations. Because the computer can process so much data without fatigue or distraction, the computational neural network is able to find subtle patterns in large data sets in a short amount of time. And, because the computational neural network operates on digital data, it is able to make quantitative associations and estimate physical properties or characteristics of the target.

The most interesting problems in geophysical data interpretation are difficult. Difficult problems require creative solutions. Creative problem solving is more likely to occur by drawing on the varied backgrounds and experiences of a team than on a solitary person with a single expertise. One of the aspects of neural computing that I find most fascinating is the eclectic nature of the field and the researchers past and present. We don't often appreciate how a particular research field is shaped by the backgrounds of the seminal contributors. Nor do we appreciate how the disparate fields of philosophy, cognitive psychology, neurophysiology, and mathematics can bring to bear creative solutions to difficult geophysical problems.

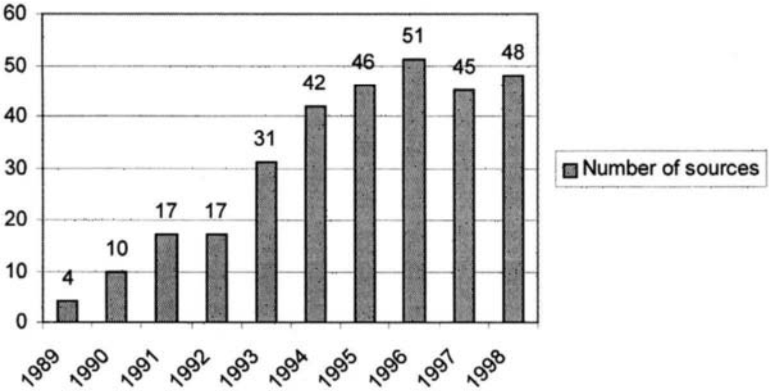


Figure 1.2. Number of different journals and conferences publishing papers on applications of computational neural networks in geophysics.

2. HISTORICAL DEVELOPMENT

Neural networks seem to have appeared out of the blue in the late 1980s. In fact, we can trace the foundations of computational neural networks back nearly a century. James Anderson and Edward Rosenfeld edited a compendium of the seminal papers in the development of computational neural networks (Anderson and Rosenfeld, 1988) for those interested in looking at some of the original derivations of neural network algorithms. The history of neural network development that I describe in the following passages draws heavily from Anderson and Rosenfeld.

The first steps on the development path of computational neural networks were taken by the eminent psychologist William James (1890) at the end of the 19th century. James' work was significant in that he was the first to discuss the memory functions of the brain as having some understandable, predictable, and perhaps fundamentally simple structure. While James' teachings about the brain's function do not mention mathematical models of neural function he does formulate some general principles of association that bear a striking resemblance to the later work of Donald Hebb (1949) and others. In his classic introductory psychology textbook, Psychology (Briefer Course), James did not present the brain as a mysterious, hopelessly complex, infinitely capable entity. Rather, he points out repeatedly that the brain is constructed to survive, not necessarily think abstractly. "It has many of the characteristics of a good engineering solution applied to a mental operation: do as good a job as you can, cheaply, and with what you can obtain easily" (Anderson and Rosenfeld, 1988). The human brain has evolved in this particular world with specific requirements for survival. In other words, the functionality of the brain is species dependent because of the different requirements species have of the world. Being able to recognize patterns, form concepts, and make associations has had far more impact on our survival than being able to solve complex arithmetic equations in our heads. Many of the computational neural networks we will discuss in this book share similar traits: they are poor at what we consider to be simple arithmetic but excel at complex associative problems.

The fundamental question being asked by psychologists during the late 19th and early 20th century was how, given thought A, the brain immediately came up with thought B? Why did a particular sound or smell or sight always invoke a certain thought or memory? The answer lies in associative memory. James (1890) writes, "...there is no other elementary causal law of association than the law of neural habit. All the materials of our thought are due to the way in which one elementary process of the cerebral hemispheres tends to excite whatever other elementary process it may have excited at any former time." Furthermore, "The amount of activity at any given point in the brain cortex is the sum of tendencies of all other points to discharge into it, such tendencies being proportionate (1) to the number of times the excitement of each other point may have accompanied that of the point in question; (2) to the intensity of such excitements; and (3) to the absence of any rival point functionally disconnected with the first point, into which the discharges might be diverted."

James (1890) continues to discuss association in the context of recall - total and partial. That is, how a "going" sequence of thoughts may evoke a "coming" sequence of secondary thoughts. I have to learn the names of a large number of students every semester. If I meet a student on the street a few semesters after having them in class, I may not be able to immediately recall the name. I may, however, remember where they sat in class, who they sat next to, the group project they worked on, names of students in that group, etc. Eventually, enough of these memories will bring back the name of the student. If I had total recall, the sequence of thought I just described would do more than bring back the name of one student, it would bring back the entire content of a long train of experiences. Rather than total recall, I exhibit partial recall. As James (1890) states, "In no revival of a past experience are all the items of our thought equally operative in determining what the next thought shall be. Always some ingredient is prepotent over the rest. The prepotent items are those which appeal most to our *interest*." An object of representation does not remain intact very long in our conscience. Rather it tends to decay or erode. Those parts of the object in which we possess an interest resist erosion. I remember a student's name because it is of interest. I do not remember the clothes the student wore or a million other details because those objects were not of interest and hence eroded. "Habit, recency, vividness, and emotional congruity are all reasons why one representation rather than another should be awakened by the interesting portion of a departing thought."

Partial recall gives way to focalized recall in which the similarity of objects evokes the thought. We see a reflection pattern in a seismic section that reminds us of a pattern we saw in a well log. The well log pattern reminds us of a portion of a magnetic profile we processed years ago. There is no physical relationship between the patterns but memory of one pattern helps retrieve similar patterns. Focalized recall happens quickly and is not as guided as the voluntary recall described below.

The above discussions would lead one to believe that the process of suggestion of one object by another is spontaneous, our thoughts wandering here and there. In the case of reverie or musing this may be true. A great deal of the time, however, our thoughts are guided by a distinct purpose and the course of our ideas is voluntary. Take the case of trying to recall something that you have temporarily forgotten. You vaguely recall where you were and what you were doing when it last occurred. You recollect the general subject to which it pertains. But the details do not come together so you keep running through them in your mind. From each detail there radiate lines of association forming so many tentative guesses. Many of these are seen to be irrelevant, void of interest and therefore discarded. Others are associated with other details and those associations make you feel as if you are getting close to the object of thought. These associations remain in your interest. You may remember that you heard a joke at a friend's house. The friend was Tom. The occasion was Sally's birthday party. The conversation centered on aging. The joke was about memory. You remember the punch line and finally you remember the joke. The train of thought just described was voluntary. You controlled the sequence because you had a goal for the thoughts.

James (1890) concludes that, "...the difference between the three kinds of association reduces itself to a simple difference in the amount of that portion of the nerve-tract supporting the going thought which is operative in calling up the thought which comes. But the modus operandi of this active part is the same, be it large or be it small. The items constituting the coming object waken in every instance because their nerve-tracts once were excited continuously with those of the going object or its operative part. This ultimate physiological law of habit among the neural elements is what runs the train."

I briefly summarized James (1890) because it is an early and interesting example of analyzing a problem, in this case association, and then relating it in terms of neural connections - training if you will. The work of James (1890) leads rather nicely into that of McCulloch and Pitts (1943). Whereas James (1890) postulated the idea of neural excitement, Warren McCulloch and Walter Pitts formalized it mathematically.

2.1. McCulloch and Pitts neuron

Warren McCulloch was one of those eclectic researchers I mentioned earlier. McCulloch came from a family of doctors, lawyers, engineers and theologians and was himself destined to enter the ministry. In 1917, after his first year at Haverford College, the eminent Quaker philosopher, Rufus Jones, asked him what he intended to do with his life. McCulloch answered that he did not know but there was a question he would like to answer: "What is a number that a man might know it and what is a man, that he might know a number?" (McCulloch, 1965). McCulloch joined the Naval Reserves during World War I where he taught celestial navigation at Yale and worked on the problem of submarine listening. He stayed at Yale to get his undergraduate degree in philosophy with a minor in psychology. At Columbia he received his M.A. degree in psychology and then went on to medical school to study the physiology of the nervous system. After working at Bellvue Hospital and Rockland State Hospital for the Insane, on the nature of schizophrenia, he went back to Yale to work with noted psychiatrist Dusser de Barenne on experimental epistemology in psychology. In 1941 he joined the faculty at the University of Illinois as a professor of psychiatry and started working with a graduate student named Walter Pitts in the area of mathematical biophysics related to the nervous system (McCulloch, 1965). Together McCulloch and Pitts set forth in their 1943 paper "The logical calculus of ideas immanent in nervous activity" to describe for the first time how the behavior of any brain could be characterized by the computation of mathematical functions. McCulloch moved on to the Research Laboratory of Electronics at MIT in 1952 where he worked on the circuit theory of brains and on nuclear magnetic resonance imaging.

The McCulloch - Pitts neuron is governed by five assumptions.

- The neuron is a binary device. Input values to the neuron can only be 0 or 1.
- Each neuron has a fixed threshold. The threshold is the numerical value the sum of the inputs must exceed before the neuron can calculate an output. The threshold is usually set equal to 1.
- The neuron can receive inputs from excitatory connection weights (w=+1). It can also receive inputs from inhibitory connection weights (w=-1), whose action prevents a neuron from turning on.
- There is a time quantum for integration of synaptic inputs. During the time quantum, the neuron responds to the activity of its weights. We call this synchronous learning because all of the inputs must be present before the state of the neuron can be updated.
- If no inhibitory weights are active, the neuron adds its inputs and checks to see if the sum meets or exceeds its threshold. If it does, the neuron becomes active.

Figure 1.3 shows an example of a McCulloch-Pitts neuron. We have a simple unit with two excitatory inputs, *A* and *B* and with a threshold of 1. A weight connected to an active unit outputs a 1. At t=0, if *A* and *B* are both inactive then at t=1 the unit is inactive. If at t=0, *A* was active and *B* was inactive then at t=1 the unit would be active. This unit is performing the logical operation *INCLUSIVE OR*. It becomes active only if *A* OR *B* OR BOTH *A* AND *B* are active.

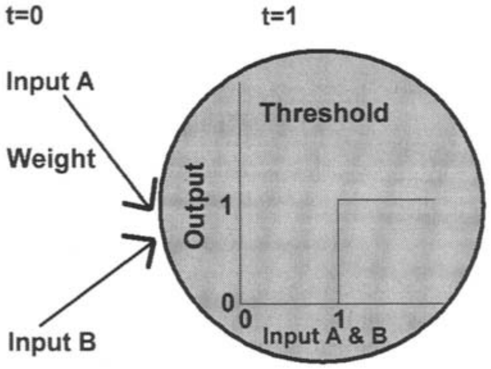


Figure 1.3. Schematic of a simple McCulloch-Pitts neuron that performs logic calculations using constraints based on known neurophysiology at the time.

The McCulloch-Pitts neuron is a simple threshold logic unit. The authors represented their unit as a proposition. The network of connections between the simple propositions was capable of creating very complex propositions. McCulloch and Pitts showed that their neuron model could compute any finite logical expression. This in turn suggested that the brain was potentially a powerful logic and computational device since the McCulloch-Pitts neuron was based on what was known about neurophysiology at the time. One of the most revolutionary outcomes of the McCulloch and Pitts paper was the notion that a single neuron was simple, and that the computational power came because simple neurons were embedded in an interacting nervous system. We know now that the McCulloch-Pitts neuron does not accurately model a neuron but their paper represents the first true connectionist model with simple computing elements connected by variable strength weights. Equations (1.1) and (1.2) in Section 2.3 represent the McCulloch-Pitts neuron that we use today.

2.2. Hebbian learning

Donald O. Hebb made the next contribution and perhaps the first that truly helped direct the future of computational intelligence. The works of both McCulloch and Hebb were strongly influenced by the study of mental illness and brain injury. Milner (1993) wrote a brief biographical article about Hebb eight years after his death that takes a fascinating look at how the twists and turns of fate led Hebb to his groundbreaking research on the relationship between neurophysiology and behavior. What follows is summarized from Milner (1993). Hebb grew up in a family of physicians but was resistant to following his siblings into the family profession. Instead, he started his professional career as an aspiring novelist and sometimes schoolteacher in the late 1920s. Knowledge of psychology is useful both to a novelist and a teacher so Hebb decided to pursue graduate studies in psychology at McGill University, working on the nature-nuture controversy and Pavlovian conditioning. A serious illness and the untimely death of his young wife left Hebb bedridden and searching for new directions and a new career. One of his thesis examiners had worked with Pavlov in St. Petersburg and recommended Hebb gain some experience in the laboratory using the Pavlovian technique. Hebb became disenchanted with the Pavlovian techniques and soon left McGill to work with Karl Lashley at the University of Chicago and later at Harvard. With Lashley, Hebb set to work on a study of how early experiences affected the vision development of rats. Hebb received his Ph.D. from Harvard for that research but jobs in physiological psychology were scarce during the Depression. By coincidence, in 1937 Hebb's sister was completing her Ph.D. in physiology at McGill and knew of a surgeon on the faculty looking for a researcher to study the affects of brain surgery on behavior. After fruitful years researching brain damage and later as a faculty member at Queens University researching intelligence, Hebb developed the theory that adult intelligence was crucially influenced by experiences during infancy. While we may accept that idea today, in 1940 it was too advanced for most psychologists. In 1942 Hebb rejoined Lashley's team, then studying primate behavior in Florida and how brain lesions affect behavior and personality. His close observations of chimpanzees and porpoises led him to the observation that play provides a good index of intelligence. Hebb was beginning work on how the brain learns to group patterns in the late 1940s. For instance, how do we recognize a piece of furniture as a chair when no two chairs we see stimulate the same nerve cells in the eye or brain. Guided by his years of diverse research and a recent discovery by noted neurophysiologist Rafael Lorente de No of feedback mechanisms in biological neural networks, Hebb was able to postulate a new theory of learning.

Hebb's great contribution is now known as "Hebbian Learning". In his 1949 book The Organization of Behavior he described the inter-relation between neurons that takes place during learning. "If the axon of an input neuron is near enough to excite a target neuron, and if it persistently takes part in firing the target neuron, some growth process takes place in one or both cells to increase the efficiency of the input neuron's stimulation" (Hebb, 1949). While Hebb never defined this relationship mathematically, we use it in most computational neural networks as the basic structure of using weighted connections to define the relationship between processing elements in a network. It was Hebb who coined the term "connectionism" that we often use to distinguish computational neural networks from other types of computational intelligence.

Hebb's theory was tested by computer simulation by Rochester et al. (1956) at IBM. This paper marked a major milestone in neural network research since proposed theories could now be rigorously tested on a computer. The availability of the digital computer both influenced development of computational neural networks and also was influenced by the research on neural networks. John von Neumann had followed the work of McCulloch and Pitts and in the publication where he first laid out the idea of a program stored in the memory of a computer, he draws parallels between the functions of the McCulloch-Pitts neuron, namely temporal summation, thresholds, and relative inhibition, and the operation of a vacuum tube (Anderson and Rosenfeld, 1988). In his book The Computer and the Brain (1958) published posthumously, von Neumann discussed the role of memory and how biological neural networks can form memories by strengthening synaptic connections to create a physical change in the brain. He also pointed out that biological neural networks cannot have a precision of any more than two to three bits. Yet, even with this very low precision, very complex operations can be reliably carried out in the brain. von Neumann concluded that we must be careful about analogies between the computer and brain because clearly the kinds of computations performed by the brain are due to the physical structure of biological neurons. Computer chips are not silicon neurons.

2.3. Neurocomputing

The decade between 1946 and 1957 witnessed the birth of neurocomputers and a split between neural network research and "artificial intelligence". Marvin Minsky, a young graduate student at Princeton constructed the first neurocomputer called the Stochastic Neural-Analog Reinforcement Computer (SNARC) in 1951 (Minsky, 1954). The SNARC, assembled in part from scavenged aircraft parts, consisted of 40 electronic "neurons" connected by adjustable links. The SNARC learned by making small adjustments to the voltage and polarity of the links (Minsky and Papert, 1988). The SNARC's contribution to neural network computing was the design of a neurocomputer rather than any interesting problems it solved. For the next decade much of the neural network research was done with special purpose mechanical devices designed to function as neurocomputers.

In the summer of 1956 John McCarthy (creator of the LISP language), then a math professor at Dartmouth, had received funding from the Rockefeller Foundation for a two-month study of the nascent field of machine intelligence. "The Dartmouth summer research project on artificial intelligence," as the conference was named, was the origination for the term "artificial intelligence". Minsky and John McCarthy went on to found the Artificial Intelligence Laboratory at MIT. A division was beginning to form at this time between researchers who pursued symbolic processing on digital computers to simulate higher-order thinking (e.g. Samuelson's checker game research) and those who believed that understanding the basic neural processes that lead to all thought and reasoning was the best approach. The various aspects of machine intelligence, be it data mining, robotic control, neural networks, natural language processing, etc., are becoming re-united today under the heading of computational intelligence. While each specialization has its own lexicon and depth of literature, there is less competitiveness or jealousy between fields as practitioners view the techniques as tools to solve pieces of complicated problems.

While Minsky demonstrated that a network using the principles of Hebbian learning could be implemented as a machine, the SNARC did not develop any new theories about learning. That breakthrough came in 1958 when psychologist Frank Rosenblatt and engineer Charles Wightman developed the Mark I Perceptron neurocomputer. With a new learning algorithm, a mathematical foundation, and both psychological and neurological fidelity, the Mark I Perceptron was able to produce behaviors of interest to psychologists, recognize patterns, and make associations. Rosenblatt did not believe that using a neurocomputer to solve the logic problems vis a viz the McCulloch-Pitts neuron as appropriate since the brain was most adept at pattern recognition and association problems, not logic problems.

2.4. Perceptron

Rosenblatt (1958) used the visual system to draw the vocabulary for his Perceptron since he was primarily interested in problems of perception. The original Perceptron consisted of three layers: an input layer of "retinal" units; a middle layer of "association" units, and an output layer of "response" units. Each layer was connected to the others by a set of randomized connections that were modified during training by a reinforcement mechanism. The middle layer of association units, however, was more like the input layer of a back-propagation network rather than a hidden layer. The layer of retinal units was more like an input buffer that reads an input pattern. The Perceptron used "winner take all" learning so that only a single unit in the response layer could be active at any time.

The patterns the Perceptron classified were binary value vectors and in the supervised mode the output classes were also binary vectors. The network was limited to two layers of processing units with a single layer of adaptive weights between them. Additional layers could be added but would not adapt. Figure 1.4 is the basic processing structure of the Perceptron.

Inputs arrive from the retinal layer, and each incoming interconnection had an associated weight w_{ji}. The Perceptron processing unit j performed a weighted sum of its input values for a pattern p of the form:

$$Sum_{jp} = \sum_{i=1}^{n} x_i w_{ji} \tag{1.1}$$

where w_{ji} was the weight associated with the connection to processing unit j from processing unit i and x_i was the value output by input unit i. We will ignore the p subscript in subsequent equations since all of the calculations are for individual patterns.

The sum was taken over all of the units i that were input to the processing unit j. The Perceptron tested whether the weighted sum was above or below a threshold value, using the rule:

$$\begin{aligned} &\mathit{if}\ Sum_j > 0\ \mathit{then}\ o_j = 1 \\ &\mathit{if}\ Sum_j \leq 0\ \mathit{then}\ o_j = 0 \end{aligned} \tag{1.2}$$

where o_j was the output value of processing unit j. The result of equation (1.2) became the output value for the network. The error was computed as the difference between the desired and calculated responses,

$$E = (d_j - o_j), \tag{1.3}$$

where d_j was the desired value for output unit j after presentation of a pattern and o_j was the output value produced by output unit j after presentation of a pattern. Since the Perceptron used only 0 or 1 for its units, the result of equation (1.3) was zero if the target and output were equal, and +1, or -1 if they were different.

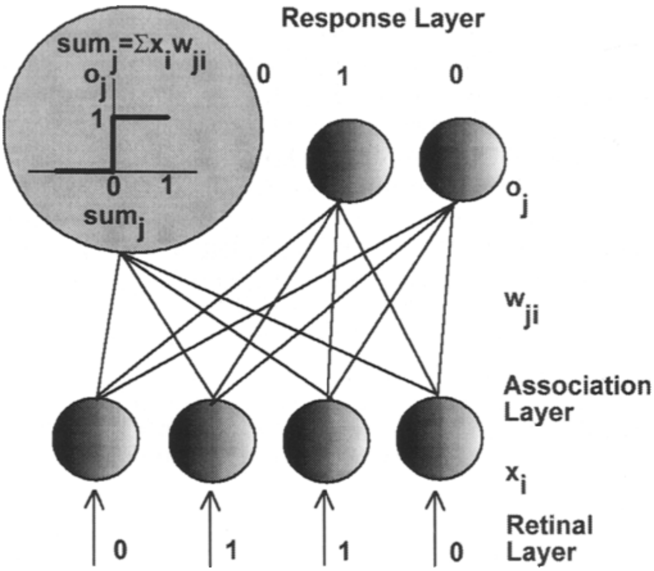


Figure 1.4. The Perceptron received binary input patterns from the retinal layer and passed them on to the association layer. A weighted sum was computed between the association and response layers. The response layer used a winner-take-all approach so only one unit was allowed to have a non-zero output.

A constant was added or subtracted from the appropriate weights during the update cycle:

$$w_{ji}^{new} = w_{ji}^{old} + \eta (d_j - o_j) x_i \tag{1.4}$$

where η is the learning rate $(d_j - o_j)$ is 1 if d_j is 1 and o_j is 0; 0 if d_j equals o_j ; -1 if d_j is 0 and o_j is 1. x_i is 1 or 0, the value of input unit i. Connection weights could only be changed if the "neurons" or processing elements connected to the output had a value of 1 and the calculated output did not match the desired output.

Since the Perceptron's memory was distributed among connection weights, it could still function if some of those weights were removed. Rather than destroying particular memories, the Perceptron would show signs of memory degradation for all patterns. Rosenblatt (1958) was aware of some of the more serious computational limitations on the Perceptron that he felt would be difficult to solve. Perceptrons can only classify linearly separable classes. Classes that can separated by a straight line in a plane are linearly separated. While it is easy to discern if a problem is linearly separable if it can be plotted in two dimensions, it is not as easy to determine in higher-dimension spaces. Rosenblatt (1958) mentioned that the Perceptron acted in many ways like a brain-damaged patient; it could recognize features (color, shape, size, etc.) but had difficulty with relationships between features (e.g. "name the object to the left of the square"). Neural networks, while good at generalization or interpolation, can be poor at abstraction. After thirty years of progress, our networks can still act brain-damaged.

2.5. ADALINE

Working independently from Rosenblatt, Bernard Widrow, an electrical engineering professor at Stanford and his graduate student Ted Hoff (inventor of the microprocessor) developed a machine similar to Rosenblatt's called the ADALINE or later the MADALINE (Widrow and Hoff, 1960) with funding from the US Navy. ADALINE stood for Adaptive LInear NEtwork and MADALINE was Many Adalines. The ADALINE is familiar to us today as an adaptive filter much like those used to cancel echoes during a telephone conversation. Like the SNARC and the Mark I Perceptron, the ADALINE was a machine that used dials and toggle switches to apply inputs to the network and lights and simple meters to display the "computed" output.

The ADALINE allowed input and output values to be either +1 or -1 instead of 1 and 0 in the Perceptron. The weighted sum of the connection weights and inputs was computed as in the Perceptron (and all later neural network algorithms),

$$Sum_j = \sum_{i=1}^{n} x_i w_{ji}. \tag{1.5}$$

The Sum_j was used to test the threshold and output the value o_j if

$$o_j = \begin{Bmatrix} +1 \; if \; Sum_{jp} \geq 0 \\ -1 \; if \; Sum_{jp} < 0 \end{Bmatrix} \quad . \tag{1.6}$$

Where the ADALINE really diverged from the Perceptron was in the weight update equation. Rather than using the thresholded output value **o** to compute the error, the weighted sum Sum_j was used:

$$E_p = (d_j - Sum_j) \text{, and} \tag{1.7}$$

$$w_{ji}^{new} = w_{ji}^{old} + \eta(d_j - Sum_j)x_i \tag{1.8}$$

Equation (1.8) is known as the Widrow-Hoff or Delta learning rule and can be shown to be the same as a least mean squares (LMS) method of changing connection weights to minimize the error function in equation (1.7). Unlike the Perceptron, which only changed connection weights, if the calculated output value o_j was in error and if the input value x_i was not zero, the ADALINE changed connection weights even if the output was in the correct class. Since the input was always non-zero, this never prevented the weights from changing either. So, the ADALINE could provide a faster convergence time than the Perceptron. The ADALINE also included an additional input called a 'bias unit' that had a constant value of 1 but a variable connection weight to the summation unit. The role of the bias was to speed the adjustment of the weighted sum to an optimal value. Widrow founded the first neurocomputer hardware company called the Memistor Corporation after the ADALINE was successfully developed and tested (Hecht-Nielsen, 1989).

The ADALINE solved many interesting problems ranging from language translation, to weather forecasting, to broom balancing on a movable cart, to finding an optimal route to back a truck up to a loading dock. Despite their interesting successes, both the ADALINE and Perceptron experienced the same failures, an inability to solve problems that were not linearly separable.

2.6. Caianiello neurons

The McCulloch-Pitts neuron, on which the Perceptron and ADALINE are based, processes static data. Caianiello (1961) proposed that the McCulloch-Pitts neuron could be modified to include time-varying sequences of data. The McCulloch-Pitts neuron became a special case of the Caianiello neuron when the time series had a length of 1. The Caianiello neuron will be described in detail in Chapter 12 where it is used as part of a seismic inversion algorithm. The basic equation,

$$Sum_j(t) = f\left(\sum_{i=1}^{N} w_{ji}(\tau)x_i(t-\tau) - \theta_j\right), \tag{1.9}$$

shows the input and output values as a function of time, t, and a neuron delay time of τ. The delay time τ has a physical basis in biological neurons as will be shown in Chapter 2. Biological neurons accumulate incoming stimuli for a period of time, τ. If a neuron-specific threshold of stimulation is not excited by time τ, the stimulation will dissipate and the neuron will remain inactive.

2.7. Limitations

In the mid 1960s, Warren McCulloch introduced two young researchers who had published papers on neural network learning theory - Marvin Minsky and Seymour Papert. Both by then were actively researching symbolic processing and artificial intelligence. In the book Perceptrons (1969) they showed that the requirement of local processing and linear threshold units in the Perceptron and ADALINE meant that these types of neurocomputers would never be able to solve simple and interesting problems like connectedness. We define a figure as connected if we can trace the entire figure without lifting the pencil off the paper.

The exclusive-or problem (XOR) is a simple two-bit parity problem that the Perceptron and ADALINE could not solve. The XOR has four input patterns as shown in Table 1.1. The XOR is not linearly separable but Minsky and Papert (1969) showed that when analyzed geometrically, the XOR is really a connectedness problem and because of the constraints imposed by local processing (only seeing one small piece of the problem at any processing unit), neural networks would not be able to solve such a problem. Rosenblatt (1958) had resisted the notion that the Perceptron computed logical functions, such as the McCulloch-Pitts neuron but Minsky and Papert (1969) showed that the Perceptron could be analyzed in terms of solving logic functions. What Minsky discovered was that Perceptron's failures had nothing to do with learning but with the relationship between architecture and the character of the problem presented to it. The trouble with Perceptrons appeared when they had no way to represent the knowledge required for solving certain problems. The moral: "One simply cannot learn enough by studying learning by itself; one also has to understand the nature of what one wants to learn. No machine can learn to recognize X unless it possesses, at least potentially, some scheme for representing X" (Minsky and Papert, 1969). Minsky and Papert (1969) concluded in their final chapter that they saw no hope in extending neural network learning theory to more complex algorithms with multiple layers. Research funding dropped dramatically after Perceptrons was published.

Table 1.1
Input and output values for the Exclusive-Or problem

Input 1	Input 2	Output
1	0	1
0	1	1
1	1	0
0	0	0

2.8. Next generation

Minsky and Papert have been given too much credit for destroying neural network research throughout the 1970s and into the 1980s. Neural network research was not healthy at the time Perceptrons was published and interest would have diminished anyway from the lack of understanding of neural processes and from the lack of adequate computational support. Time was needed to develop ideas about knowledge representation and new theories of learning. Research money became scarce and most scientists and engineers in the US who had pursued neural network research moved on to fields that could provide the funding necessary to maintain a research lab or ensure tenure status. Some researchers, mostly in cognitive psychology, continued to pursue their research and made important contributions in the 1970s. James Anderson at Brown University, Stephen Grossberg at Boston University, and John Hopfield at California Institute of Technology, all pursued different lines of research that led to major advances. Teuvo Kohonen at the Helsinki Institute of Technology also continued his research into how topological sensory maps in the brain could be recreated with a machine, leading to the famous Kohonen networks and self-organizing maps.

The resurgence of neural network research is often attributed to the publication of a nonlinear network algorithm that overcame many of the limitations of the Perceptron and ADALINE. The new network algorithm published by David Rumelhart and James McClelland (1986) was called back-propagation but in reality it was not new. The fundamentally same algorithm had been discovered by Paul Werbos during his dissertation research at Harvard and published in his thesis "Beyond Regression: New tools for prediction and analysis in the behavioral sciences" (1974). David Parker (1985) also published a similar algorithm in an MIT report. The learning rule used in back-propagation can even be traced back to the Robbins/Munro (1951) technique of finding roots of regression functions.

Perhaps the great contribution of Rumelhart and McClelland and their Parallel Distributed Processing Group at the University of California San Diego was that they published the algorithm along with a body of research supporting its applicability to interesting and complex problems in book form that was accessible to a wide range of researchers. After 1986 the interest in neural network research, as measured by numbers of publications and amount of funding, began an exponential climb. Neural networks became the overnight sensation that was 40 years in the making.

Since the publication of Parallel Distributed Processing (1986) we have seen a thousand ways to tweak the back-propagation algorithm, a plethora of algorithms related to back-propagation, and a number of fundamentally new architectures, some with real biological basis. We have seen application of neural networks in nearly every field of science and engineering, although the earth sciences have been among the slowest to explore the uses, and we have seen commercial products containing neural networks succeed on the market. Many industrial applications of neural networks can claim significant increases in productivity, reduced costs, improved quality, or new products.

The intent of this book is to present neural networks to the geophysicists as a serious computational tool - a tool with great potential and great limitations. I would like to wipe away the pejorative phrase that neural networks are a mysterious box that can not be deciphered and instead present neural networks as a tool that is very information rich. Neural networks do not suffer from a lack of information but rather an excess of information that must somehow be diffracted into simpler wavelengths that we can decipher. We need the right prism to get the information we desire. Finding that prism is no easy task.

REFERENCES

Anderson, J., and Rosenfeld, E., 1988, Neurocomputing: Foundations of Research: MIT Press.

Caianiello, E., 1961, Outline of a theory of thought-processes and thinking machines: Journal of Theoretical Biology, **2**, 204-235.

Hebb, Donald O., 1949, The Organization of Behavior: Wiley.

Hecht-Nielsen, R., 1989, Neurocomputing: Addison-Wesley.

James, William, 1890, Psychology (Briefer Course): Holt.

McCulloch, W., 1965, Embodiments of Mind: MIT Press.

McCulloch, W., and Pitts, W., 1943, A logical calculus of ideas immanent in nervous activity: Bulletin of Mathematical Biophysics, **5**, 115-133.

Minsky, M., 1954, Neural nets and the brain - model problem: Ph.D. Dissertation, Princeton University, Princeton, NJ.

Minsky, M., and Papert, S., 1969, Perceptrons: MIT Press.

Minsky, M., and Papert, S., 1988, Perceptrons, Expanded Edition: MIT Press.

Milner, P., 1993, The mind and Donald O. Hebb: Scientific American, January, 124-129.

Parker, D., 1985, Learning-logic: Technical Report TR-47, Center for Computational Research in Economics and Management Science, MIT, April.

Robbins, H., and Munro, S., 1951, A stochastic approximation method: Annals of Mathematical Statistics, **22**, 400-407.

Rochester, N., Holland, J., Haibt, L., and Duda, W., 1956, Tests on a cell assembly theory of action of the brain, using a large digital computer: IRE Transactions on Information Theory, IT-2, 80-93.

Rosenblatt, F., 1958, The Perceptron: a probabilistic model for information storage and organization in the brain: Psychological Review, **65**, 386-408.

Rumelhart, D., and McClelland, J., 1986, Parallel Distributed Processing: Explorations in the Microstructure of Cognition: MIT Press.

von Neuman, J., 1958, The Computer and the Brain: Yale University Press.

Werbos, P., 1974, Beyond regression: New tools for prediction and analysis in the behavioral sciences: Ph.D. Dissertation, Applied Math, Harvard University, Cambridge, MA.

Widrow, B., and Hoff, M., 1960, Adaptive switching circuits: IRE WESCON Convention Record, 96-104.

Chapter 2
Biological versus Computational Neural Networks

Mary M. Poulton

1. COMPUTATIONAL NEURAL NETWORKS

Computational neural network vocabulary draws heavily on cognitive psychology and neurophysiology. Networks are trained not programmed. They learn. They generalize. They can become paralyzed. They can become over specialized. The vocabulary is very qualitative for a fundamentally quantitative technique. But the vocabulary also serves to distinguish computational neural networks from mathematical algorithms such as regression or from statistical techniques and reinforces the biological and psychological foundation of the field.

All neural networks have at least three components in common - the neuron, node, or processing element (PE), the connection weight, and discrete layers that contain the PEs and are connected by the weights (Figure 2.1). The PE is the basic computational unit in a network and is classified according to its role in the network. A PE that receives information only from an external source, an input file for example, is called an input PE. Input PEs may scale the incoming values before passing them on but other than that they perform no computation. A PE that passes its computed values to an external source, an output file for example, is called an output PE. The output PEs also compute the error values for networks performing supervised learning (learning in which a desired output value is provided by the operator). Any PE that is not in an input or output layer is referred to as a hidden PE. The term hidden is used because these PEs have no direct connection to the external world. In a biological model, input PEs would be analogous to sensory neurons in our eyes, ears, nose, or skin; output PEs would be motor neurons that cause muscles to move; hidden PEs would be all the remaining neurons in the brain and nervous system that process the sensory input.

2. BIOLOGICAL NEURAL NETWORKS

Before moving on to the other parts of a network, it is worth spending some time explaining how a biological neuron generates a signal that can be transmitted to other neurons. We borrow more vocabulary from the neurophysiologists when we explain the internal workings of a computational neural network. Neurons generate electrical signals that are frequency-coded rather than amplitude coded. Hence our brains are more FM than AM. The cell body of a neuron is called the soma and it contains a solution that is richer in potassium ions than sodium ions. The exterior fluid surrounding the cell is an order of magnitude richer in sodium ions than potassium ions. Hence, a potential difference of approximately 70 millivolts is created across the cell membrane by the concentration gradient. The interior of the cell is negative with respect to the exterior. (Fischbach, 1992). Any change in the concentration gradient will change the potential difference and generate an "activation" of the neuron if a "threshold" potential difference is exceeded.

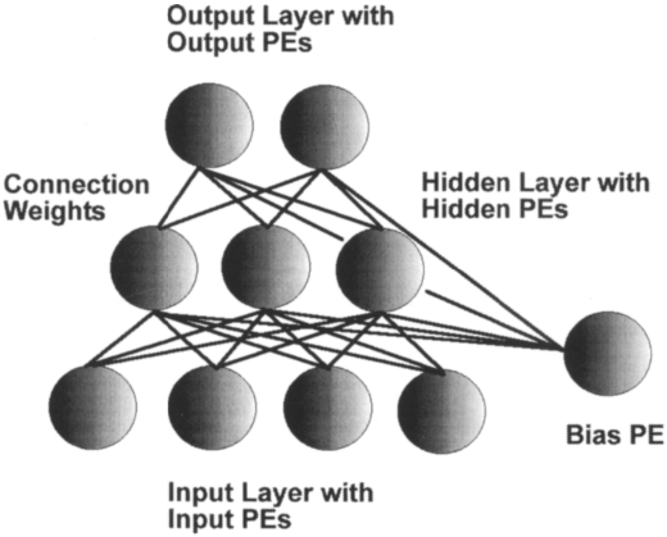


Figure 2.1. Parts of a computational neural network. The diagram shows a fully-connected, feed-forward network sometimes called a Multi-Layer Perceptron.

The cell membrane contains bi-polar molecules that have a phosphoric acid head, which is attracted to water, and a glyceride tail that is repelled by water. The molecules align themselves in the membrane with their heads pointed outward and their tails pointed inward. The tails form a barrier to the ions of salts that are contained within the cell. To change the concentration gradient, however, we need a mechanism to transport the potassium and sodium ions across the membrane. Embedded in the polarized cell membrane are proteins that serve as gates or "channels" and pumps. When the voltage difference across the membrane is locally lowered, the channels open and sodium ions pour into the cell. The local change in concentration gradient causes more channels to open and the previous ones to close, thus propagating an electrical signal down the axon at a rate of nearly 300 km / hour. A neuron may be able to discharge 50 to 100 voltage spikes each second so the time between spikes is around 10 to 20 milliseconds.

To restore the cell to equilibrium, protein "pumps" are activated which can exchange the sodium ions for potassium ones. Operating at a maximum rate, a pump can exchange 200 sodium and 130 potassium ions each second. A small neuron may have a million pumps, so we have a maximum capacity of nearly 200 million sodium ions per second per neuron. A neuron can return to its resting potential in approximately 3 to 5 milliseconds.

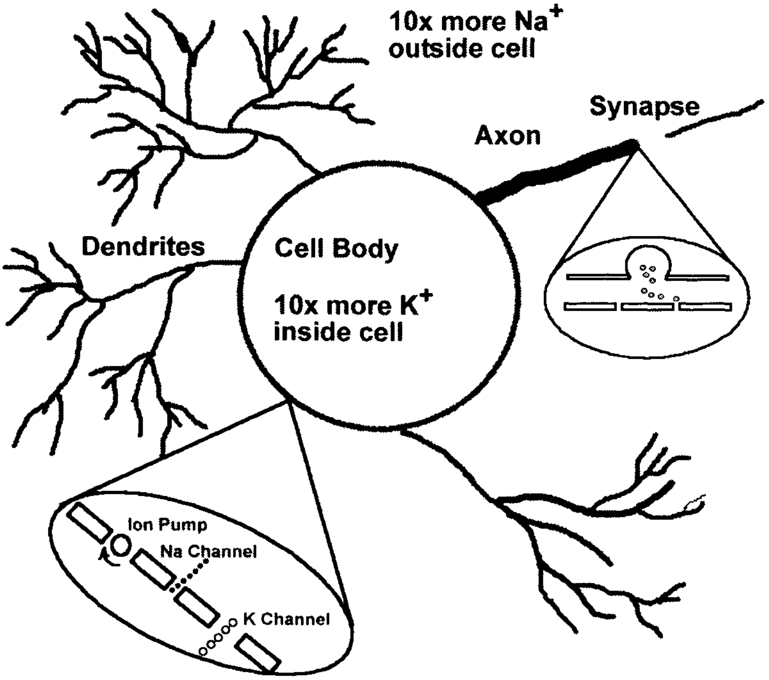


Figure 2.2. A simplified neuron in a biological neural network. The sodium and potassium channels selectively allow specific ions to enter or leave the cell body while the ion pump exchanges the sodium potassium to maintain equilibrium. At the synapse, the synaptic vesicle releases neurotransmitters into the synaptic cleft where they activate dendrites attached to the synapse.

Whether a signal gets propagated down the axon of a neuron depends on whether the "activation" potential difference exceeds the "threshold" of the cell. The neuron can sum and integrate the incoming signals, basically allowing them to accumulate over a short period of time. If the arriving signals are of high enough frequency (spaced closely in time), then there is little time for a cell to return to its resting state and the potential for exceeding the threshold is high. The process of summing incoming signals and checking a threshold is the fundamental operation of a McCulloch-Pitts neuron. Incorporating a time constant into the summing process is the basis of the Caianiello neuron. Unlike the biological neuron, computational PEs act on amplitude modulation rather than frequency modulation.

The activation of the neuron is the potential difference that is achieved after summing the incoming signals. A generalized graph of the firing rate of a typical biological neuron as a function of the input current is shown in Figure 2.3a. The neuron does not fire until a certain threshold is reached and beyond a certain input current, it saturates and the firing rate does not increase. The threshold function is non-linear and is more sensitive to activations within a middle range of values. The typical threshold function used in neural networks is also shown in Figure 2.3b. The function generating this threshold is the logistic function that is sigmoidal in shape and usually referred to as the sigmoid function. The mathematical requirements for the threshold function are described in Chapter 3.

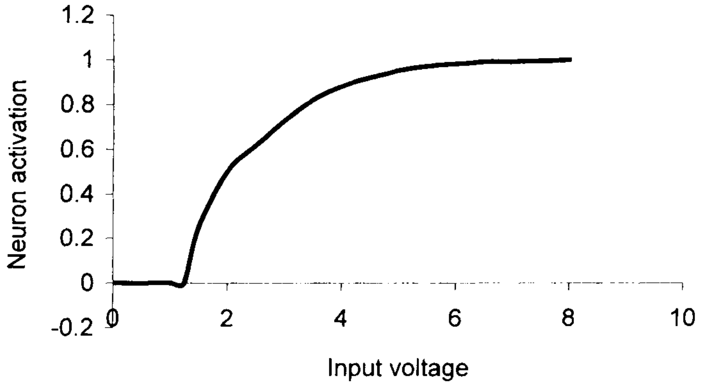


Figure 2.3a. A generalized graph of the relationship between the input current and neuron firing rate shows the "sigmoidal" shape used as the threshold function in biological neurons.

The neuron in figure 2.2 includes a synapse. The synapse is the location in a biological neuron that allows a signal to be transferred from one neuron to several others. The electrical impulse that travels down the axon is converted back to a chemical signal at the synapse. The voltage received at the synapse causes a small "sack" called the synaptic vesicle to merge with the presynaptic membrane and release transmitter molecules. The transmitter molecules travel across the gap of 20 to 50 nm between the synapse and the receiver membrane of another neuron. Once at the new neuron, or post-synaptic membrane, the transmitter molecules function as a lock and key mechanism whereby only certain molecules can attach to the post-synaptic membrane and start to generate a new electrical signal.

The transmitter molecules or neurotransmitters can either excite the next neuron to fire or can inhibit it from firing. Some of the better known neurotransmitters include serotonin (on which the anti-depressant Prozac operates), acetocholine (which is activated by nicotine), dopamine (which is activated by cocaine), and gamma aminobutyric acid (GABA) (which is acted on by valium). Computational neural networks have not exploited the variety of behaviors that a biological synapse can produce. The connection weight is the computational equivalent of the synapse. While the value of the connection weight can have the effect of stimulating or inhibiting the connected PE to exceed its threshold, the connection weight does not carry with it a classification that could create different behaviors (i.e. similar to having only one type of neurotransmitter).

Neurons in a biological nervous system exist in small neighborhoods where the type of neuron and the type of behavior produced are similar. For example, the pre-frontal cortex behind your forehead is responsible for manipulating symbolic information and short-term memory while the cerebellum at the base of your skull helps control balance and movement.

The outer layer of the brain is called the cerebral cortex and is divided into four layers each with its own type of neurons. Each layer is vertically connected. Different regions of the cortex perform specialized tasks such as speaking, processing visual information, understanding speech, discrimination of shape and texture, etc. Hence, in a biological nervous system we have a network of networks, each performing some specialized process and then transmitting the results to other networks. Within each network we have layers where even more specialized processing is performed and transmitted vertically as well as horizontally through the network. In a computational neural network we typically work with just one network, although modular or hierarchical architectures are becoming more popular, and within that network we have connected layers of PEs.

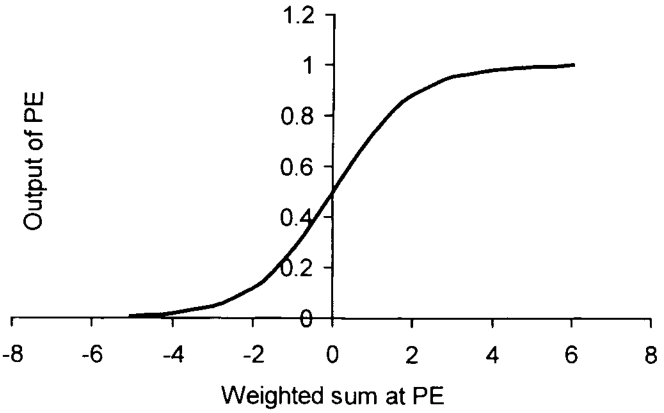


Figure 2.3b. Typical sigmoidal-shaped threshold function used in a computational neural network.

3. EVOLUTION OF THE COMPUTATIONAL NEURAL NETWORK

Much of the major development in computational neural networks is still driven in part by a desire to more closely emulate biological neural networks. Most new architectures are developed with some biological or psychological fidelity. Those of us who are primarily interested in computational networks as a tool tend to focus on ways to increase the speed and efficiency of existing algorithms.

Table 2.1 shows the evolution of the computational neural network as knowledge of neurophysiology and cognitive psychology has progressed. We continue to learn a great deal about the specific roles different neurotransmitters play in learning and behavior at the neuron level. The advent of PET scans has led to new understanding of larger-scale activation of neural networks in different regions of the brain. As we continue to learn more and more about how the small-scale chemistry of the brain impacts the larger-scale issues of learning and behavior, we find new ways to model the processes with computers. However, we should keep in mind John von Neumann's quote that "the logic of the brain is not the language of mathematics" (von Neumann, 1958). As the Nobel-prize winning physicist, and later neural network researcher, Leon Cooper points out "for animals there need be no clear separation between memory and 'logic' ...the mapping A can have the properties of a memory that is non-local, content addressable and in which 'logic' is a result of association and an outcome of the nature of the memory itself" (Cooper, 1973). In other words, the brain does not process and store information following the same elegant rules of logic and mathematics we have imposed on computers. Logic is an outcome of the memory associations we make rather than the cause of the memory associations.

To some researchers, our inability to impose the same rules of mathematics on a neural network, be it biological or computational, as on a mathematical algorithm, makes the biologically based system flawed. If a solution to a problem cannot be described in rigorous mathematical terms then the solution is suspect. The field of computational neural networks tries to walk the fine line between preserving the richness and complexity of the biological associative memory model while using the language and logic of mathematics.

Table 2.1
Impact of neurophysiological developments and advances in cognitive science on development of computational neural networks.

Year	Advance in biological / psychological understanding of brain	Contribution to computational neural networks
1943	Mathematical description of a neuron	McCulloch-Pitts Neuron
1949	Formulation of learning mechanism in the brain	Hebbian learning
1958	Connectionist theories of sensory physiology	Perceptron
1973	Cortical physiology	Self Organizing Maps, Adaptive Resonance Theory
1977	Speech perception	Bi-directional associative memories
1981	Use of non-linear threshold similar to neural activation function	Back-propagation
1987	Early visual systems	Computer chip-based networks
1991	Visual perception	Hierarchical / modular networks

REFERENCES

Cooper, L., 1973, A possible organization of animal memory and learning, *in* B. Lundquist, B., and Lundquist, S., Eds., Proceedings of the Nobel Symposium on Collective Properties of Physical Systems: Academic Press, 252-264.

Fischbach, G., 1992, Mind and Brain: Scientific American, **267**, 48-59.

von Neumann, J., 1958, The Computer and the Brain: Yale University Press.

This Page Intentionally Left Blank

Chapter 3
Multi-Layer Perceptrons and Back-Propagation Learning

Mary M. Poulton

1. VOCABULARY

The intent of this chapter is to provide the reader with the basic vocabulary used to describe neural networks, especially the multi-layer Perceptron architecture (MLP) using the back-propagation learning algorithm, and to provide a description of the variables the user can control during training of a neural network. A more detailed explanation of the mathematical underpinnings of many of the variables and their significance can be found in Bishop (1995) or Masters (1993).

Networks are described by their architecture and their learning algorithm. The architecture is described by the fundamental components of PEs, connection weights and layers and the way each component interacts with the others. A connection strategy refers to the way layers and PEs within layers are connected. A **feed-forward** strategy means that layers are connected to other layers in one direction, from the input to the output layer. A **feed-back** strategy means that some or all of the layers have changeable connections that go back to a previous layer (e.g. from the first hidden layer to the input layer). A **fully-connected** strategy means that every PE in a layer is connected to every PE in another layer. A **pruning** strategy means that PEs are selectively disconnected from each other. An **interconnected** strategy means that PEs within a layer are connected to each other. Within an interconnected strategy the connected PEs may "**compete**" (i.e. winner take all strategy) with each other so that only one can be active or they can **cooperate** so that several are active. Networks can be **heteroassociative** when the output pattern vector is different from the input. The network is **autoassociative** if the input and output pattern vectors are the same. Autoassociative networks are useful for pattern completion and for compression.

The learning strategy refers to how the network is trained. In **supervised** learning you must provide input / output pairs. The output patterns that you provide are compared to the output that the network computed and any difference between the two must be accounted for by changing parameters in the network. In the simplest implementation of a network, the least-means square rule or delta rule is used to update the parameters:

$$parameter_{new} = parameter_{old} + 2\eta\varepsilon x, \tag{3.1}$$

where η is a positive constant, ε is an error term, and x is an input value.

In **unsupervised** learning the network is provided only with input patterns and it finds common features in groups of those patterns. Supervised and unsupervised learning are analogous to the supervised and unsupervised classification routines like Maximum Likelihood and K-means clustering. Unsupervised learning is discussed in more detail in Chapter 5.

2. BACK-PROPAGATION

We will start by examining a feed-forward, fully connected multi-layer Perceptron using the back-propagation learning algorithm. Most authors simplify the name of this type of network to back-propagation since it is the most commonly used architecture for that learning algorithm. Back-error propagation or back-propagation is the most widely used type of neural network today. It is very robust and easy to implement and use. The method was first invented by Paul Werbos in 1974 in his Ph.D. dissertation in the social sciences at Harvard and later reinvented by David Parker in 1985 and presented to wide readership by David Rumelhart and James McClelland of the PDP group at San Diego in 1986.

In its most common configuration, the back-propagation network has three layers: an input layer, a single hidden layer, and an output layer. An additional PE called a bias unit is also included. We will talk more about the bias unit in Section 3.6.

The first layer receives the input from the training file. Each input pattern contains a fixed number of input elements or input PEs. The output layer receives a fixed number of output PEs. The number of input and output PEs is dictated by the problem being solved. We only supply output PEs for supervised training. There are a fixed number of training patterns in the training file. Figure 3.1 shows a scatter plot where the points have been classified into one of three possible classes. The class boundaries are drawn as straight lines to help separate the points on the plot. The classes have been assigned a binary code that can be used for network training. Figure 3.2 shows the corresponding training file for Figure 3.1 with two input values representing x- and y-coordinate values and five output values representing five possible classes for the input data points. The output coding is referred to as "1-of-n" coding since only one bit is active for any pattern. Together one input pattern and output pattern constitute one training pattern in the training set. The MLP architecture to solve the classification problem in Figures 3.1 and 3.2 is shown in Figure 3.3.

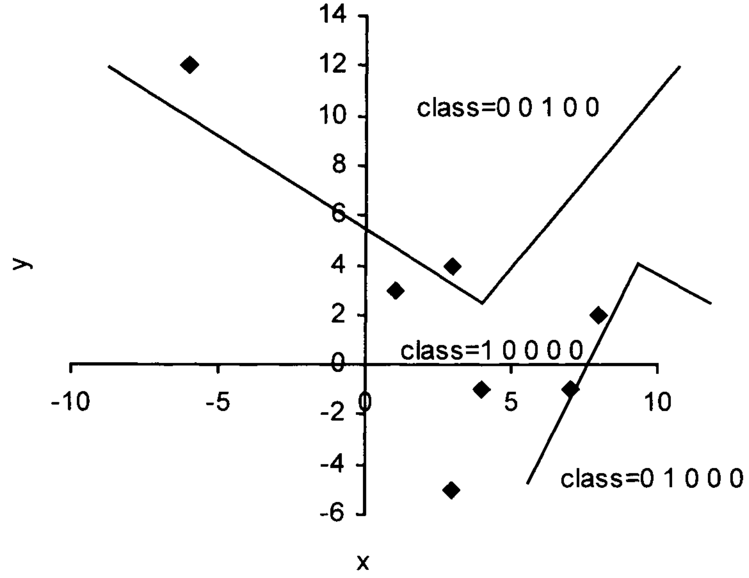
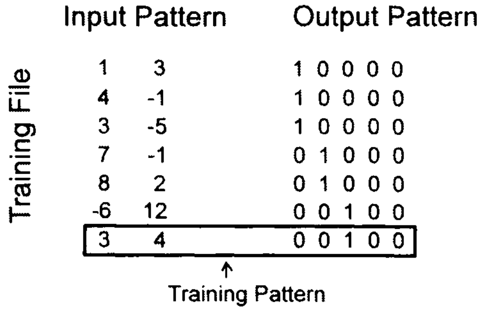


Figure 3.1. A scatterplot of data points and their corresponding classification. The data can be used as input to a neural network for supervised training. The class boundaries indicate that the problem is not linearly separable and therefore appropriate for a non-linear network such as back-propagation.



Training File

Input Pattern	Output Pattern
1 3	1 0 0 0 0
4 -1	1 0 0 0 0
3 -5	1 0 0 0 0
7 -1	0 1 0 0 0
8 2	0 1 0 0 0
-6 12	0 0 1 0 0
3 4	0 0 1 0 0

Training Pattern

Figure 3.2. A sample training file. The input pattern is represented by x- and y-coordinates of data points. The corresponding output pattern is a classification of the datum location.

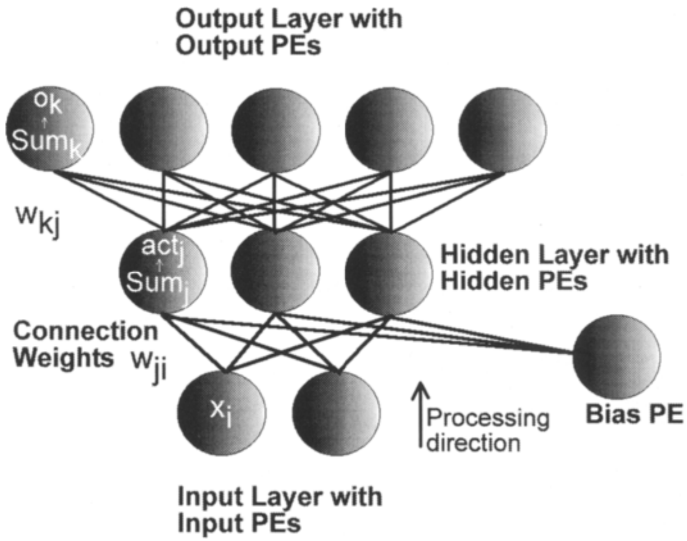

Output Layer with Output PEs
ok
Sumk
wkj
actj
Sumj
Hidden Layer with Hidden PEs
Connection Weights wji
xi
Processing direction
Bias PE
Input Layer with Input PEs

Figure 3.3. A feed-forward multi-layer Perceptron architecture typically used with a back-propagation learning algorithm. The configuration shown here corresponds to the data set shown in Figure 3.2.

The value received by each PE is multiplied by an associated connection weight:

$$Sum_j = \sum_{i=1}^{n} w_{ji} x_i + w_{jb}, \tag{3.2}$$

where Sum_j represents the weighted sum for a PE in the hidden layer. The connection weight vector $\vec{w}$ represents all the connections weights between a PE in the hidden layer and all of the input PEs. The input vector $\vec{x}$ contains one element for each value in the training set. The bias unit is added as the connection weight w_{jb} since the bias has a constant input value of 1.0. The role of the bias unit is described in Section 3.6. In subsequent equations the bias connection weights will be assumed to be part of the weight vector $\vec{x}$ and will not be shown as a separate term.

At a PE in the next or hidden layer all products are summed. This is called the output of the PE. The output is passed through a **threshold function** and this becomes the **activation** (act_j) of the PE

$$act_j = f_j(Sum_j) \tag{3.3}$$

Activation is a term drawn from the neurophysiology lexicon and refers to the state of a biological neuron becoming physically active if the incoming electrical stimulus exceeds the threshold of the cell. The threshold function is typically represented by the sigmoid function,

$$f_j(Sum_j) = \frac{1}{1+e^{-Sum_j}}, \tag{3.4}$$

or by the hyperbolic tangent function,

$$f_j(Sum_j) = \frac{e^{Sum_j} - e^{-Sum_j}}{e^{Sum_j} + e^{-Sum_j}}. \tag{3.5}$$

For function mapping applications we may use a linear threshold function for the output layer PEs. The linear output function is the identity function $f(x)=x$.

The activation is multiplied by the connection weights going to the next layer. The input signal is propagated through the net in this way until it reaches the output layer,

$$Sum_k = \sum_{j=1}^{m} w_{kj} act_j + w_{kb}, \tag{3.6}$$

$$o_k = f_k(Sum_k). \tag{3.7}$$

The connection weights in the net start with random values so after the first iteration, the calculated output will not match the desired output given in the training file. The net needs some form of error correction. The best start (when we don't know any better) is a modified mean-squared error,

$$e_p = \frac{1}{2}\sum_{k=1}^{L}(d_{pk} - o_{pk})^2. \tag{3.8}$$

The error e_p for a pattern p is calculated as the square of the difference between the output values given in the training file, d_{pk} , and those calculated by the network, o_{pk} , for each element of the output vector. Since the training goal is to minimize the error with respect to the connection weights, we must take the derivative of equation (3.8). For the sake of simplicity I will drop the negative signs associated with the error gradients in the following equations:

$$\frac{\partial e_p}{\partial w_{kj}} = \frac{\partial e_p}{\partial Sum_{pk}} \frac{\partial Sum_{pk}}{\partial w_{kj}} \quad . \tag{3.9}$$

We can solve equation (3.9) by looking at each component individually.

$$\frac{\partial Sum_{pk}}{\partial w_{kj}} = \frac{\partial}{\partial w_{kj}} \sum_{k} w_{kj} act_{pj} = act_{pj}. \tag{3.10}$$

If we introduce a new variable,

$$\delta_{pk} = \frac{\partial e_p}{\partial Sum_{pk}}, \tag{3.11}$$

to represent the derivative of the error with respect to the sum, equation (3.9) becomes

$$\frac{\partial e_p}{\partial w_{kj}} = \delta_{pk} act_{pj}. \tag{3.12}$$

Rewriting equation (3.11) gives us

$$\frac{\partial e_p}{\partial Sum_{pk}} = \frac{\partial e_p}{\partial o_{pk}} \frac{\partial o_{pk}}{\partial Sum_{pk}}. \tag{3.13}$$

Each component of equation (3.13) can be solved:

$$\frac{\partial o_{pk}}{\partial Sum_{pk}} = f_k^{'}(Sum_{pk}), \tag{3.14}$$

and

$$\frac{\partial e_p}{\partial o_{pk}} = (d_{pk} - o_{pk}). \tag{3.15}$$

Substituting equations (3.14) and (3.15) into equation (3.13) results in

$$\delta_{pk} = (d_{pk} - o_{pk}) f_k^{'}(Sum_{pk}). \tag{3.16}$$

The weight changes we make for the connections between the hidden and output layer, based on minimization of the error function in equation (3.8) are found by substituting equation (3.16) into equation (3.12),

$$\Delta w_{kj} = \frac{\partial e_p}{\partial w_{kj}} = (d_{pk} - o_{pk}) f_k^{'}(Sum_{pk}) act_{pj}. \tag{3.17}$$

Before we can adjust the weights connected to the output layer we need to know the error attributed to each PE in the hidden layer. We had the advantage for the output PEs that we knew the error based on the training set values. We do not know what the output values should be for the hidden PEs, so we need to express the relationship between the calculated values for each output PE and the activation of the PEs in the hidden layer as

$$\frac{\partial e_p}{\partial w_{ji}} = \frac{1}{2}\sum_k \frac{\partial}{\partial w_{ji}}(d_{pk} - o_{pk})^2. \tag{3.18}$$

Expanding the derivative on the right side of equation (3.18) results in

$$\frac{\partial e_p}{\partial w_{ji}} = \sum_k (d_{pk} - o_{pk}) \frac{\partial o_{pk}}{\partial Sum_{pk}} \frac{\partial Sum_{pk}}{\partial act_{pj}} \frac{\partial act_{pj}}{\partial Sum_{pj}} \frac{\partial Sum_{pj}}{\partial w_{ji}}. \tag{3.19}$$

Once again we can solve each of the components in equation (3.19) and substitute back into the equation. We know from equation (3.14) that

$$\frac{\partial o_{pk}}{\partial Sum_{pk}} = f_k^{'}(Sum_{pk}). \tag{3.20}$$

$$\frac{\partial Sum_{pk}}{\partial act_{pj}} = \frac{\partial}{\partial act_{pj}} \sum_k w_{kj} act_{pj} = w_{kj}. \tag{3.21}$$

$$\frac{\partial act_{pj}}{\partial Sum_{pj}} = f_j^{'}(Sum_{pj}). \tag{3.22}$$

$$\frac{\partial Sum_{pj}}{\partial w_{ji}} = \frac{\partial(\sum w_{ji} x_{pi})}{\partial w_{ji}} = x_{pi}. \tag{3.23}$$

Substituting equations (3.20) through (3.23) into equation (3.19) results in

$$\frac{\partial e_p}{\partial w_{ji}} = \sum_k (d_{pk} - o_{pk}) f_k^{'}(Sum_{pk}) w_{kj} f_j^{'}(Sum_{pj}) x_{pi}. \tag{3.24}$$

The first terms after the summation are the same as equation (3.16) so we can simplify equation (3.24)

$$\frac{\partial e_p}{\partial w_{ji}} = (f_j^{'}(Sum_{pj}) \sum_k \delta_{pk} w_{kj}) x_{pi}. \tag{3.25}$$

We can further simplify equation (3.25) by defining the variable δ_{pj} as

$$\delta_{pj} = f_j^{'}(Sum_{pj}) \sum_k \delta_{pk} w_{kj}. \tag{3.26}$$

So, the weight changes that are made on the connections between the hidden and input layers are a function of the error terms for the output PEs and results in

$$\frac{\partial e_p}{\partial w_{ji}} = \Delta w_{ji} = \delta_{pj} x_{pi}. \tag{3.27}$$

Once we know the errors we need a method or **learning rule** to change the weights in proportion (η) to the error. The delta rule in equation (3.1) is applied to the output layer PEs for each input pattern,

$$\Delta w_{kj} = \eta \delta_{pk} act_{pj}, \tag{3.28}$$

so the new connections weights between the hidden and output layers take on the values of

$$w_{kj}^{new} = w_{kj}^{old} + \eta \delta_{pk} act_{pj}, \tag{3.29}$$

and the connections weights between the input and hidden layers become

$$w_{ji}^{new} = w_{ji}^{old} + \eta \delta_{pj} x_{pi}. \tag{3.30}$$

After the weights are changed we re-apply the inputs and repeat the whole process. The cycle of applying an input, calculating an output, computing an error and changing the weights constitutes one **iteration** of the network. Alternatively, the error values can be summed over the duration of a training **epoch**. An epoch is equal to one presentation each for every training sample (i.e. one pass through the training set). The weights are not changed until all the training patterns have been presented once. This method of training is usually referred to as **batch learning**. There is usually no practical advantage to batch learning over updating the weights after each pattern presentation.

Learning stops when the error drops below a specified tolerance or the network reaches a user-specified number of iterations. After learning stops, the weights are frozen and the network performance can be **validated** using new data. The validation process consists of presenting a new set of input data to the network and calculating the output values. The network does not make any weight change during the testing or validation process. Validation is often distinguished from **testing** the network. Testing results can be used to improve the network training and can be done during the training process. Validation is used to prove the network can solve the problem for which it was trained. Validation data should not be used as part of the training process.

By using equations (3.2) through (3.7) and the weight update equations (3.29) and (3.30) we can program the simplest version of the back-propagation neural network. The connection weights start with random values, usually uniformly distributed between user-specified values (frequently -0.1 and +0.1). The numbers of input and output PEs are fixed by the particular application. The number of hidden PEs must be specified. The user specifies the value for η that remains fixed throughout the training process. Training stops after a user-specified number of iterations or the error drops below a specified threshold value.

The equations outlined for the back-propagation network represent a gradient descent technique and the neural network is prone to the same problems that any algorithm using this technique would experience. Convergence to a global minimum is not guaranteed; learning can be very slow; and connections can take on extreme values paralyzing the learning process. The deficiencies in the back-propagation algorithm have been addressed by hundreds of researchers resulting in a multitude of improvements and variations to the algorithm. We will look at some of these improvements in the context of the variables the user typically specifies in the network. Alternatives to gradient descent are discussed in Chapter 5.

3. PARAMETERS

The user has control over several parameters in the back-propagation network:

1. Number of layers
2. Number of hidden PEs
3. Threshold function (logistic, tanh, linear, sine, etc.)
4. Weight initialization
5. Learning rate and momentum
6. Bias
7. Error accumulation
8. Error calculation
9. Regularization and weight decay

Another important parameter, which the user may or may not have control over, is the number of training examples.

In the following sections we will use an example of training an MLP using back-propagation learning to calculate the value of the sine function given a noisy input value. The noise (in degrees) is calculated as

$$\cos(200x - 4x^2 + 5x^3)/100. \tag{3.31}$$

The training file consists of a single input of the sine function plus the noise function and a single output of the corresponding sine function at 0.5-degree intervals from 0.5 to 360 degrees (Figure 3.4). The associated test set contains noisy and clean sine function data from 360.5 to 720 degrees at 0.5-degree intervals. While the clean sine function data is the same in the training and test sets, the noise is not. The network must operate as a filter by learning a function that represents the noise added to the sine function.

3.1. Number of hidden layers

Several researchers give guidance on the number of hidden layers required to solve arbitrary problems with a back-propagation-type network. The Kolmogorov's Mapping Neural Network Existence Theorem (Bishop, 1995, Hecht-Nielsen, 1990) says that a neural network with two hidden layers can solve any continuous mapping $y(x)$ when the activation of the first hidden layer PEs is given by monotonic functions $h(x_i)$ and the activation of the second hidden layer PEs is given by

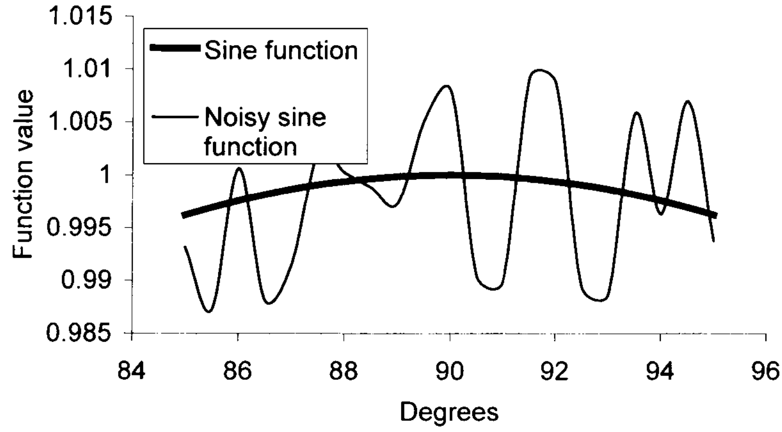


Figure 3.4. A portion of the sine function and sine function with noise used for training the network. The network receives the noisy sine function as input.

$$z_j = \sum_{i=1}^{n} \lambda_i h_j(x_i). \tag{3.32}$$

The constant λ, analogous to the step size η, is real and ranges from 0 to 1.The output of the network is then

$$y_k = \sum_{j=1}^{2n+1} g_k(z_j). \tag{3.33}$$

The first hidden layer has *n* PEs, the second layer has 2*n*+1 PEs. The functions g_k k=1,2,..,m are real and continuous.

Kolmogorov's Theorem is important because it shows at least theoretically that a neural network architecture similar to the MLP can solve any continuous mapping problem. While the theorem is interesting, it has conditions that limit its practical application. We do not know what functions to use for *h* and *g*. The theorem assumes *h* is a non-smooth function that in practice makes the network overly sensitive to noise in the input data. Finally, the theorem assumes a fixed number of hidden PEs with variable activation functions *h* and *g*. In practice, neural networks have a variable number of hidden PEs with known activation functions.

Many researchers (Cybenko, 1989; Hornik et al., 1989; Hecht-Nielsen, 1990) tackled the problem of proving how many hidden layers are sufficient to solve continuous mapping problems. The theoretical proofs all must make certain assumptions that make practical implementation difficult or unrealistic. Bishop (1995) provides a more practical proof that a network with one hidden layer using a sigmoidal activation function can approximate any continuous function given a sufficient number of hidden PEs. Summarizing from Bishop

(1995), suppose we want to estimate the function $y(x_1,x_2)$ given the input variables x_1,and x_2. We can approximate this function through Fourier decomposition as,

$$y(x_1,x_2) \cong \sum_n a_n(x_1)\cos(nx_2), \tag{3.34}$$

where the a_n coefficients are functions of x_1. The coefficients can also be described by a Fourier series

$$y(x_1,x_2) \cong \sum_n \sum_k a_{n\,k} \cos(kx_1)\cos(nx_2). \tag{3.35}$$

If we define a variable $z_{nk}=kx_1+nx_2$ and $z'_{nk}=kx_1-nx_2$ and replace the two cosine terms in equation (3.35) with their trigonometric identity, we can write a new equation

$$y(x_1,x_2) \cong \sum\sum a_{nk}(\frac{1}{2}\cos(z_{nk}) + \frac{1}{2}\cos(z_{nk}')). \tag{3.36}$$

The cos(z) function (or any function f(z)) can be represented by a piecewise continuous function of the form

$$f(z) \cong f_0 + \sum_{i=0}^{N} \{f_{i+1} - f_i\} h(z - z_i), \tag{3.37}$$

where h is the Heavyside function. So, the desired function $y(x_1,x_2)$ can be approximated by a series of step functions which are represented by sigmoidal threshold functions in the network. The accuracy of the approximation will be determined by the network architecture and training parameters.

The conclusion most researchers have drawn is that one hidden layer is sufficient but including an additional hidden layer may, in some cases, improve the accuracy and decrease the learning time.

For our sample problem of filtering a noisy sine function, Figure 3.5 shows that the RMS error for the test set is large when we have a small number of PEs in hidden layer 1 regardless of how many PEs are in hidden layer 2. As the number of PEs in hidden layer 1 increases the RMS error decreases. Adding a second hidden layer does not decrease the RMS test error in this example. As the number of PEs in hidden layer 1 approaches 10, the RMS error increases, indicating a possible over fitting of the training data as described in the next section.

3.2. Number of hidden PEs

While the number of hidden PEs plays an important role in the accuracy of a neural network, the importance of finding an absolute optimum number is often over emphasized. The input data representation and training set design are often far more critical than the number of hidden PEs in controlling the accuracy of the results.

Network performance can be considered a quadratic function of the number of hidden PEs so a decrease in number could result in increased performance, as could an increase. Figure 3.6 shows a general relationship between the overall error of the network and the number of PEs in the first hidden layer. The minimum error can be very broad, allowing for a range in the number of hidden PEs that can solve the problem.

Some researchers have suggested that the geometric mean of the inputs and outputs is a good predictor of the optimum number of hidden PEs in networks with fewer output nodes than inputs. Too few hidden PEs and the network can't make adequately complex decision boundaries. Too many and it may memorize the training set. Two excellent sources of information on understanding the role of the hidden PE are Geman et al. (1992) and Lapedes and Farber (1988).

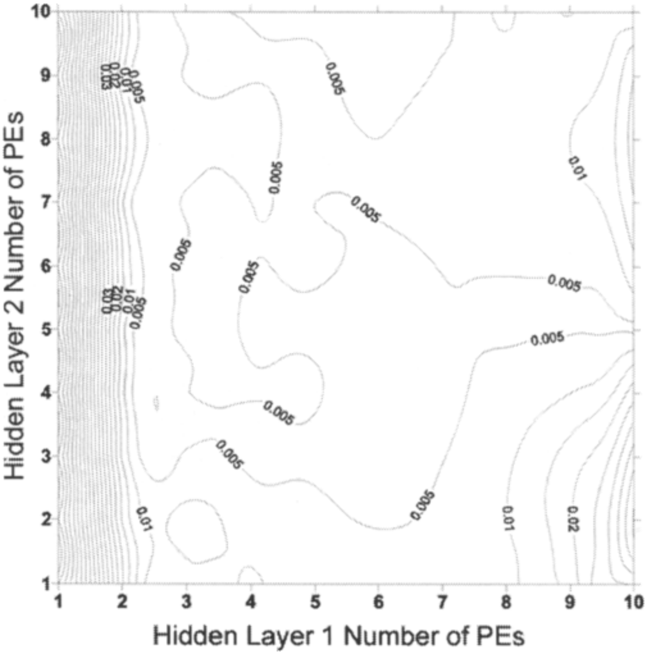


Figure 3.5. RMS test error for the sine function estimation as a function of number of hidden PEs in the first and second hidden layers.

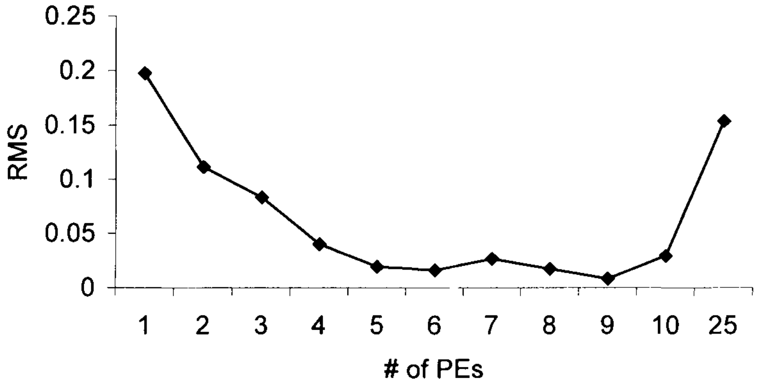


Figure 3.6. The relationship between RMS error on training data as a function of the number of hidden PEs in a single hidden layer for the noisy sine function data set.

We can rewrite the equation for the output from the forward pass through the back-propagation network as a function of the input and hidden layers:

$$o_k = \sum_j w_{kj} f_j \left(\sum_i w_{ji} x_{ip} + \Theta_j \right) + \Theta_k, \tag{3.38}$$

where Θ is the bias connected to each hidden and output PE. If more than one hidden layer is desired then equation (3.38) can easily be expanded to accommodate it. The value of writing the network output in this form is that once training is complete and the values for the connections weights are set, we can have the output as a function of the input values. If the output is one-dimensional and the input is two dimensional or less then we can easily plot the functional relationship. Lapedes and Farber (1988) showed that with a sigmoidal transfer function equation (3.38) forms a sigmoidal surface, a "ridge", whose orientation is controlled by the values of the connection weights between the input and hidden layer; position is controlled by the values of the bias weights connected to the hidden layer; height is controlled by the weights between the hidden and output layer. If a second hidden layer is used, the connection weights for a second function can be superimposed on the first function but with a different orientation to form a "bump". Hence, with two hidden layers the back-propagation network is able to approximate functions in a manner analogous to a Fourier Transform. In the neural network case, however, the "bumps" used to approximate the function are not restricted to trigonometric functions.

We can also use equation (3.38) to see how the outputs of the hidden layer change during training. Figure 3.7 shows the initial state of the network trained to output the value of the sine function given noisy input data. The initial weights have random values. After 1,000 iterations, the connection weights between the input and hidden layer have taken on values that allow the network to reproduce the desired function, albeit with different magnitudes

(Figure 3.8). The role of the connection weights between the hidden and output layers is to scale the function to the proper magnitude. By the time the weights have been updated 100,000 times the output layer nearly matches the desired function values (Figure 3.9). While the rms error continues to improve slightly after 100,000 iterations, the connection weights do not change much.

Geman et al. (1992) takes the approach that in any non-parametric inference technique the estimation error can be decomposed into a bias component and a variance component. An error with a large bias component indicates inaccurate models. Using model-free learning leads to high variance. In the case of neural network design and training, the bias / variance dilemma posed by Geman et al. (1992) means that a network that has too many hidden PEs or that is trained too long will have a large bias component of the error. A network with a high bias component of the error will fit noise or very fine structure in the data leading to poor validation results when new data are presented. A network with too few PEs or that is undertrained will produce a very smooth fit to the data and will produce poor validation results as well. The approach proposed by Geman et al. (1992) involves calculating both the bias and variance components of the mean-squared error during training. As the number of hidden PEs is increased, the bias component should decrease and the variance component should increase. The minimum bias and the maximum variance should correlate with the optimum number of hidden PEs. If the number of hidden PEs are held constant and the bias and variance are plotted as a function of training iterations, the trend will be for variance to increase and bias to decrease as the number of training iterations increases. The training error will often continue to decrease as training continues.

If the sole criterion for stopping training is the error on the training data then the network may produce poor validation results because of the tendency to over fit noise in the data. In other words, the variance component of the error increases if we train too long. Hence, better results are often obtained by terminating training before the network converges to a global minimum. The method Geman et al. (1992) propose for computing the bias and variance errors during training is very time consuming. A faster approach to determine when to stop training can be to periodically interrupt training by testing the network on another data set (not the validation set). When the error on the test data begins to increase, training should be stopped. When we are estimating a function value, such as the sine function example, we may not observe an increase in the testing error over time. In that situation the decision to terminate training is based on whether the accuracy of the function estimation after a certain training interval is sufficient for our application based on an error analysis of the test data. In some cases, such as the problem presented in Chapter 9, it may be difficult to quantify the test error because of the nature of the problem or number of processing steps involved to produce the actual test result. In such cases, training is usually stopped when some measure of the training error stops improving.

The arbitrariness of the number of hidden PEs and the need for trial and error design was addressed by Fahlman and Lebiere (1990) who developed the cascade correlation network which starts with no hidden layers and trains to its optimum performance. After the network has trained for a user-specified number of iterations, a hidden PE is added. Each new PE is fully connected to the input and output layer and also receives input from all previous hidden PEs. Training starts again for the same number of iterations. When a new PE is added the connection weights attached to the previous PEs are held fixed and do not train. At the end of this training session, test data are presented to the network and the RMS error is computed and compared against the error from the previous trial. If the error has improved another hidden PE is added. The procedure continues until the error ceases to improve.

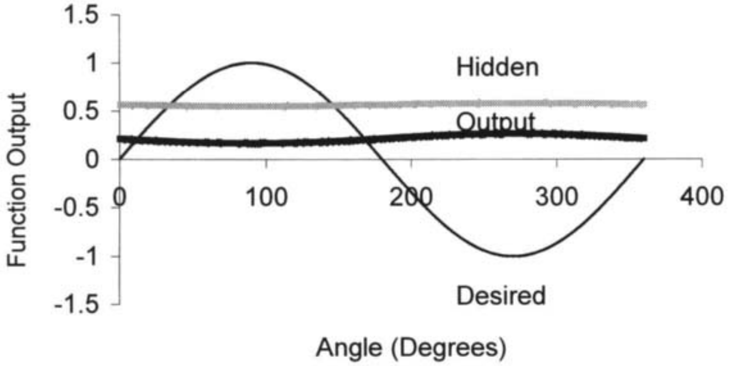


Figure 3.7. Both the hidden and output layers estimate nearly constant values for the training set before any weight adjustments are made.

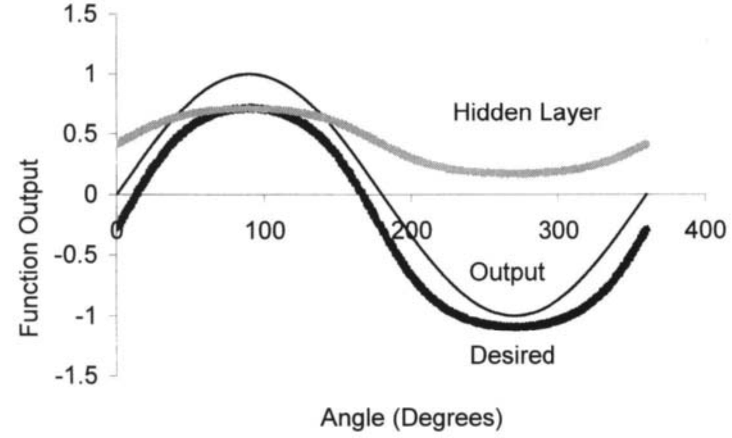


Figure 3.8. After 1,000 iterations the connection weights between the input and hidden layers have duplicated the approximate shape of the sine function while the weights between the hidden and output layer perform a scaling to the desired magnitude.

The alternative approach to building the hidden layer from an initial state with no hidden PEs is to start with a large number of hidden PEs and reduce, or prune, the number of nodes or weights over time. Several pruning techniques have been suggested in the literature. The simplest technique is to specify a threshold based on the average magnitude of all the connection weights in the network and prune those weights that are a user-specified standard deviation away from the mean. Pruning based on magnitudes is an ad hoc approach and seldom works well.

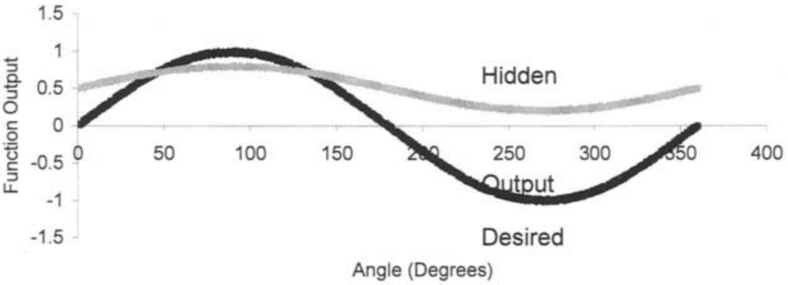


Figure 3.9. After 100,000 iterations the connection weights between the hidden and output layers have matched the desired function output while the weights between the input and hidden layers have not changed much. Little change is observed in the connection weights after 100,000 iterations.

Le Cun et al. (1990) and Hassibi and Stork (1993) proposed two different solutions to the pruning problem that are based on the use of the Hessian matrix. The overall goal is to find a fast method to compute the change in the output error when a particular weight is eliminated. Hassibi and Stork (1993) called the sensitivity of the error to a particular connection weight the "saliency of the weight". The Hessian matrix of a neural network represents the second derivatives of the error with respect to the weights:

$$H = \frac{\partial^2 e}{\partial w_{ji} \partial w_{kl}}. \tag{3.39}$$

Le Cun et al. (1990) created an "optimal brain damage" network by computing the value:

$$\frac{H_{ji} w_{ji}^2}{2}, \text{ only for j=i.} \tag{3.40}$$

for each connection weight and eliminating the weights with the smallest values. The "brain damage" approach to training assumes that the off diagonal terms of the Hessian can be ignored, which is not usually a good assumption.

Hassibi and Stork (1993) presented the "optimal brain surgeon" approach to address the shortcomings of the brain damage approach. The "brain surgeon" approach uses the inverse of the Hessian to compute the importance or "saliency" of each connection weight by

$$\frac{w^2_{ji}}{2\left[H^{-1}\right]_{ji}}, \text{ only for j=i.} \tag{3.41}$$

If the saliency is much smaller than the overall network error then that weight can be pruned and the remaining connection weights are updated according to

$$\partial w_{ji} = -\frac{w_{ji}}{\left[H^{-1}\right]_{ji}} H^{-1} b_{ji}, \tag{3.42}$$

where $\vec{b}$ is a unit vector in weight space parallel to the w_{ji} axis (Bishop, 1995).

3.3. Threshold function

Most implementations of back-propagation networks use either the sigmoid or tanh function to perform the threshold operation. As shown in Chapter 14, Table 14.2, other functions may also be used. Networks using the tanh function sometimes converge faster than the networks using the sigmoid function, but often the two functions will produce the same overall accuracy (Figure 3.10).

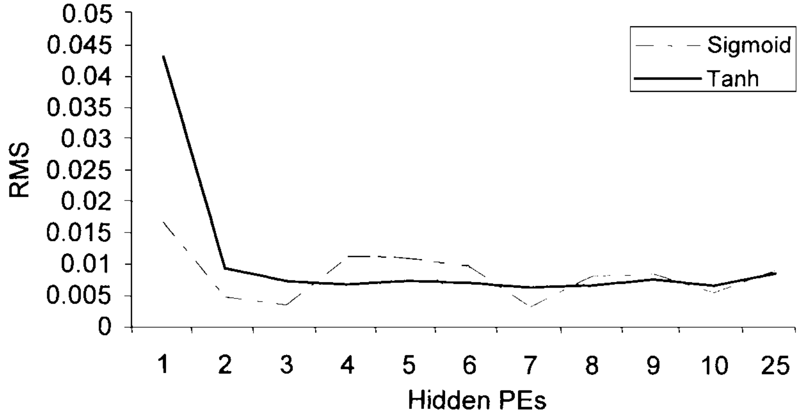


Figure 3.10. The sine function estimation network convergence rate is smoother for the tanh function than the sigmoid function.

If the network is solving a classification problem, then a sigmoid or tanh function should be applied to the output layer to force the values closer to 0 or 1. If the problem involves estimating values of a function, then a linear threshold function is usually applied to the output layer. If the output values contain noise, a tanh function often performs better than a linear output function.

3.4. Weight initialization

The size of the initial random weights is important. If they are too large, the sigmoidal activation functions will saturate from the beginning, and the system will become stuck in a local minimum or very flat plateau near the starting point. A sensible strategy is to choose the random weights so that the magnitude of the typical net input net_{pj} to unit j is less than - but not too much less than - unity. This can be achieved by taking the weights w_{ji} to be of the order of 1 / k_j where k_j is the number of input PEs (*i's*) which feed forward to the hidden PEs (*j*). Weights should usually be chosen from a uniform rather than a Gaussian distribution, although as Figure 3.11 shows, sometimes a Gaussian distribution of initial weights can result in slightly faster convergence.

A network solution to a particular problem is non-unique in the sense that many different combinations of connection weight values may lead to the same overall result. Table 3.1 shows the weight values between the input and hidden layer for our sine estimation problem for three different trials. There are no differences in the network configuration for the trials other than the initial weight values. Trial 1 represents weight values after training the network. In trial 2 the weights are re-initialized to new random values and the network is retrained with identical learning parameters to trial 1. In trial 3 the same random starting weights as used in trial 2 are used. We can see from Table 3.1 that when different random weights are used for training, the final weight values can vary a lot even when no other parameters change. When the same initial weight values are used, however, the final weight values are very similar. The weight values probably differ slightly because the input patterns are presented in random order to the network for each trial.

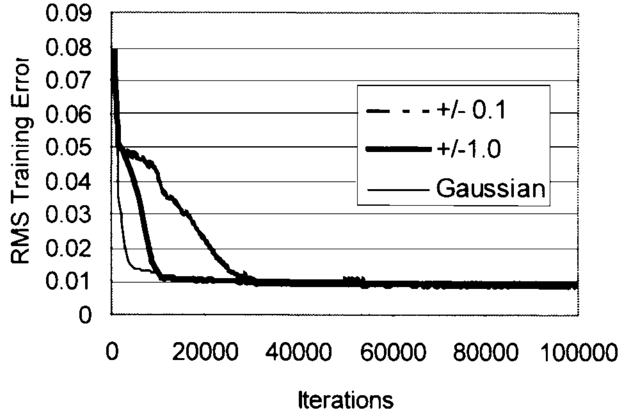


Figure 3.11. The convergence rate for the sine estimation network is fastest when the connection weights are initialized using a Gaussian distribution with a range of [-1,1]. More commonly we use a uniform distribution with a range of [-0.1, 0.1].

Table 3.1
Comparison of hidden connection weights

Trial 1	Trial 2	Trial 3
-0.09976	-0.23166	-0.28048
-0.59691	-1.37848	-1.37060
-1.33939	0.23759	0.23045
0.30361	-0.29671	-0.25760
0.11369	0.28433	0.28323
1.57277	0.24339	0.27001
-0.41932	-0.21658	-0.20101
0.25380	-0.53788	-0.47691
0.46127	-1.49012	-1.50304

3.5. Learning rate and momentum

The learning rate or step size, η, and the momentum value, α, control the amount of change made to the connection weights. Different values of the parameters are typically specified separately for each layer in the network. A schedule can also be specified that allows the parameters to vary as a function of the iteration number of the training cycle. A small value for the learning rate will slow the convergence rate but will also help ensure that the global minimum will not be missed. A larger value of the learning rate is appropriate when the error surface is relatively flat. The magnitudes of the learning rate and momentum are traded off so that a small learning rate can be coupled with a larger momentum to increase convergence speed and a larger learning rate is usually coupled with a smaller momentum to help ensure stability,

$$w_{ji}(t+1) = w_{ji}(t) + \eta \delta_{pj} x_{pi} + \alpha(w_{ji}(t) - w_{ji}(t-1)). \tag{3.43}$$

The idea is to give each connection some inertia or momentum, so that it tends to change in the direction of the average downhill force that it feels, instead of oscillating wildly with every little kick. Then the effective learn rate can be made much larger without divergent oscillations occurring.

If the error surface is relatively flat, resulting in a nearly constant value of the error derivative in weight space over time, then the weight changes will be

$$\Delta\bar{w} \approx -\frac{\eta}{1-\alpha}\frac{\partial \bar{e}}{\partial \bar{w}}, \tag{3.44}$$

with an effective learning rate of $\eta/(1-\alpha)$ (Hertz, et al., 1991) and the network can converge faster. If the error surface we are traversing is highly convoluted, then the weight changes will have a tendency to oscillate (Figure 3.12). In this case, the addition of a momentum term will tend to dampen the oscillations and again the network can converge faster.

The values for the learning rate and momentum terms are often picked by trial and error. The same values are used for all PEs in a particular layer and the values can change with time according to a user-specified schedule. Jacobs (1988) developed an algorithm called the "Delta Bar Delta" or DBD that allows a learning rate to be assigned to each connection weight and updated throughout the training process. The DBD algorithm is described in Chapter 5.

3.6. Bias

A bias unit is a PE with a constant output value of 1.0 but a trainable connection weight attached to each PE in the hidden and output layers. The bias unit was first introduced by Widrow and Hoff (1960) for the ADALINE. The bias unit had a fixed value of 1.0 but had a trainable connection weight whose magnitude served as the threshold value in equation (3.2). The bias unit is still described in many references as a threshold. Figure 2.3b shows a plot of the sigmoid function. The output value of a PE depends on the weighted sum of an input vector and a weight vector. The sum may not fall on an optimal part of the curve so one way to ensure that it does is to add a bias to the sum to shift it left or right along the curve.

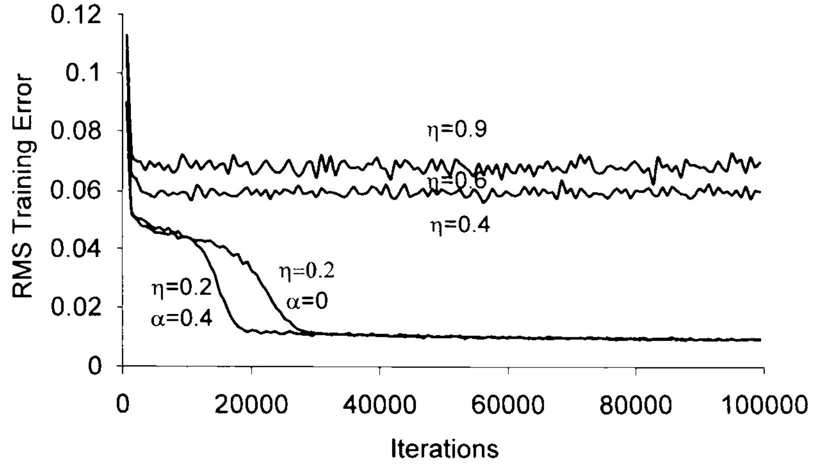


Figure 3.12. The curve labeled α=0 shows the affect of momentum when the step size is set to 0.2. The use of the momentum term improves the convergence rate although given enough time both trials converge to the same RMS error. Notice that as the learning rate becomes smaller, the error curve becomes smoother, indicating more stable training for the sine estimation problem.

The graph in Figure 3.13 compares the convergence rate of a simple back-propagation network trained on the XOR (exclusive or) problem for networks with and without a bias unit. The networks had two input PEs, four hidden PEs, and one output PE. Initial connection weights were identical. The network with a bias unit converged in less than 20,000 iterations while the network without a bias unit failed to converge even after 50,000 iterations. The exclusive or problem is a simple problem with only four training samples. For more realistic and interesting problems the affect of the bias unit on network performance is not usually this pronounced. Figure 3.14 shows the effect for our sine-estimation problem. The network with a bias PE achieves a lower RMS error for training than a network without a bias PE.

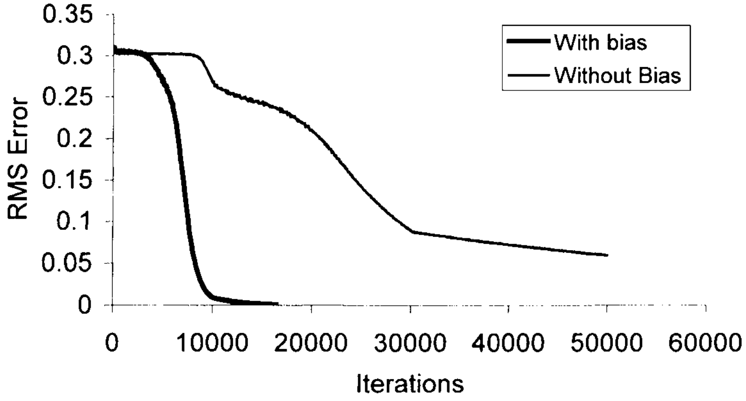


Figure 3.13. The XOR problem is solved fastest by a network using a bias element connected to each hidden and output PE.

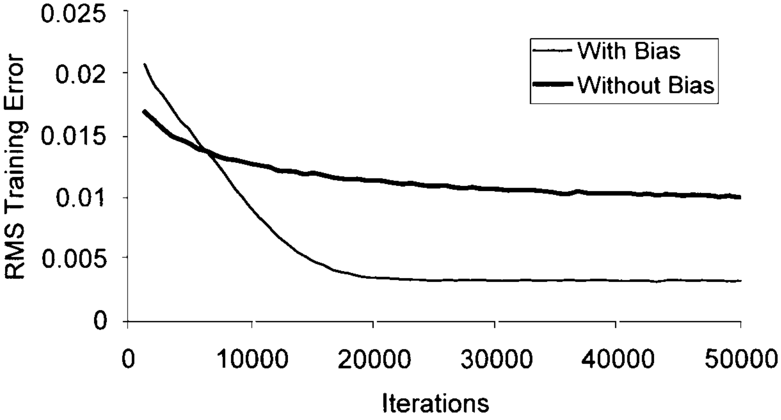


Figure 3.14. The sine function is estimated with better accuracy when a bias PE is used.

3.7. Error accumulation

Weights can either be updated after each pattern presentation or after all patterns in a training set have been presented. If the error is accumulated for all training patterns prior to changing the connection weights, we refer to the training method as batch mode. If the weights are updated after every pattern, we refer to the training as pattern mode.

The pattern mode tends to be used most often and gives good results, especially if there is some redundancy in the training data or if the training patterns are chosen in random order during training (Figure 3.15). The batch mode requires more computational overhead since the error terms must be stored for the entire training set. Both pattern and batch mode can be shown mathematically to be gradient descent techniques when equations (3.16) and (3.26) are used to update the weights. When batch mode is used to update the connection weights, the cumulative error should be normalized by the number of training samples so the value that is used as error represents the average error over the training set. Even with normalization, the average error can be large enough that unless a very small value for the learning rate is used, the network can easily become paralyzed.

Training can be performed "off line" when all the data needed for training, testing, and validation have been collected ahead of time; or it can be performed "on line" when data are collected from some process during training. On line training is most often performed in a dynamic plant environment like a refinery or an assembly line. Most network applications use off line training.

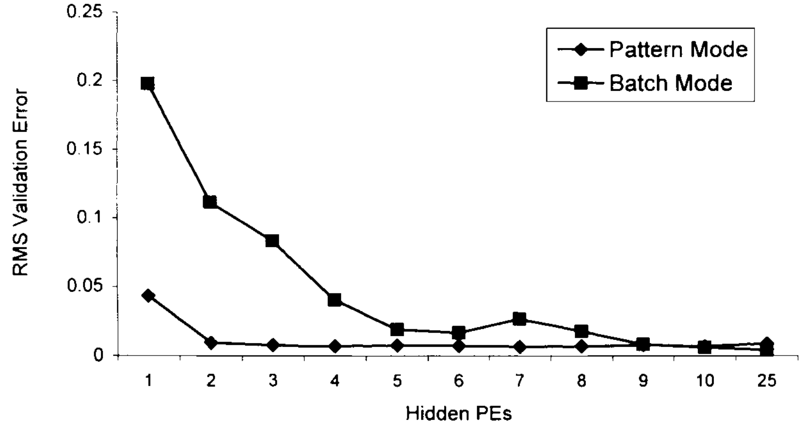


Figure 3.15. The sine estimation network achieved a lower RMS error for a smaller number of hidden PEs when error updating was performed after each pattern was presented.

3.8. Error calculation

Most neural network applications use the quadratic cost function, or mean squared error, in equation (3.8). Mean squared error is a useful cost function because large errors are weighted more heavily than small errors, thus ensuring a larger weight change. An alternative cost function is the entropic function proposed by Solla et al. (1988). The quadratic cost function tends to output a constant error term when the output of one PE saturates at the wrong extreme (Hertz et al., 1991). The entropic measure continues to learn in such cases and has been shown to solve problems that the quadratic cost function cannot (Wittner and Denker, 1988). The entropic cost function is calculated as,

$$e_p = \sum_k \{ \frac{1}{2}(1 + d_{pk}) \log \frac{1 + d_{pk}}{1 + o_{pk}} + \frac{1}{2}(1 - o_{pk}) \log \frac{1 - d_{pk}}{1 - o_{pk}} \}. \tag{3.45}$$

Differentiating the entropy equation and assuming we are using the tanh function, we get the delta weight equation as

$$\delta_{pk} = (d_{pk} - o_{pk}). \tag{3.46}$$

The main difference from the standard equation shown in equation (3.16) is that the derivative of the threshold is missing. Without the derivative, larger changes can be made in areas with a relatively flat error surface without danger of oscillating when the error surface is more convoluted.

Fahlman (1989) proposed a modification of equation (3.46) that still includes the derivative of the threshold function

$$\delta_{pk} = \{ f'_k(Sum_{pk}) + 0.1 \} (d_{pk} - o_{pk}). \tag{3.47}$$

Another approach is to change the error difference $(\vec{d} - \vec{o})$ instead of (or as well as) the derivative, increasing delta when $(\vec{d} - \vec{o})$ becomes large (Fahlman, 1989). For example,

$$\delta_{pk} = arctanh \frac{1}{2}(d_{pk} - o_{pk}). \tag{3.48}$$

3.9. Regularization and weight decay

Regularization theory allows us to transform an ill-posed problem into a well-posed problem by imposing some "smoothness" constraints (Poggio and Girosi, 1990). Regularization is probably familiar to anyone who has worked with geophysical inversion codes. Neural network applications that involve reconstructing a smooth mapping from a set of training samples are ill posed because the data are usually insufficient to completely reconstruct the mapping and because of noise in the data. Regularization theory can be applied to computational neural networks either through network design such as the use of radial basis functions (see Chapters 11 and 16) or through weight decay.

Regularization involves adding a penalty term to the error function, usually of the form (Bishop, 1995),

$$e_{pk} = e_{pk} + c\lambda, \tag{3.49}$$

where λ is the penalty term and *c* is a constant that moderates the extent to which the penalty is applied. The simplest penalty term is a weight decay of the form,

$$\lambda = \frac{\rho}{P}\sum_{l} w_l^{\,2}. \tag{3.50}$$

The parameter ρ is a user-specified smoothness parameter, *P* is the number of training samples, and w_l is the connection weight (Swingler, 1996). Since smoothness in a network is associated with small values for the connection weights, equation (3.53) works by moving the weighted sums computed at each node to the central linear portion of the activation function (Bishop, 1995). A more robust regularizer, one that is scale invariant, is given by,

$$\lambda_{jl} = \frac{\rho_1}{P}\sum_{k} w_{kj}^{\,2} + \frac{\rho_2}{P}\sum_{l} w_{jl}^{\,2}. \tag{3.51}$$

The bias weights should not be included in the regularization parameter calculations if you use the scale invariant regularization since they will distort the mean of the network output (Bishop, 1995).

4. TIME-VARYING DATA

Time series can be processed by computational neural networks either by extracting windows of the series and treating the data as a static pattern or by using an architecture called a recurrent network. As discussed in Chapters 1 and 12, a Caianiello neuron model could also be used in a MLP architecture to construct a network capable of processing time-varying data.

Windowing a time series is the easiest way to classify or predict time series values. Care must be taken in processing the input data to remove any trends or cyclical variations that are not diagnostic. Masters (1993) provides a good discussion of processing methods for time series data. As with any prediction technique, neural networks perform best if they do not have to predict events too far into the future. The disadvantage of treating a time series as a set of fixed-time windows of data is the network does not learn any relationship between the time windows.

A recurrent network architecture allows signals to feedback from any layer to a previous layer. Recurrent networks can be based on the MLP structure by feeding back the output values from the hidden layer PEs or output PEs to the input layer. In an Elman network the hidden layer values are fed back to PEs in the input layer called "context units" (Figure 3.16). The context units provide a memory or context for each input based on the activity from the previous pattern (Skapura, 1996). Context, in a recurrent architecture, prevents identical patterns, that occur at different times, from being confused with each other. A Jordan network feeds the output values back to PEs on the input layer and also interconnects the feedback PEs (Figure 3.17). The Jordan network is able to relate patterns to each other in a time sequence (Skapura, 1996).

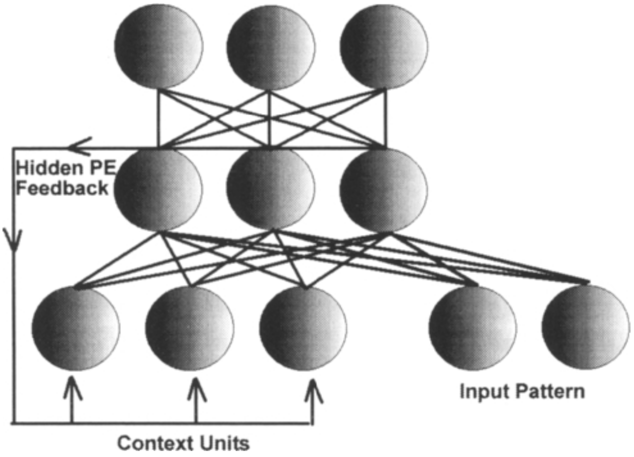

Hidden PE Feedback
Input Pattern
Context Units

Figure 3.16. An Elman network feeds the hidden PE activations to context units in the input layer.

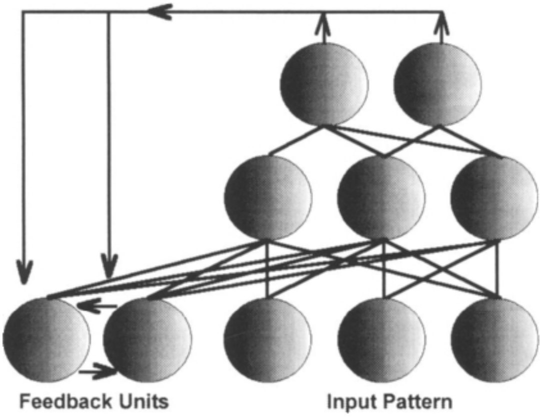

Feedback Units
Input Pattern

Figure 3.17. A Jordan network feeds the output activations to the input layer and allows interconnections between the feedback units on the input layer.

REFERENCES

Bishop, C., 1995, Neural Networks for Pattern Recognition: Oxford Press.

Cybenko, G., 1989, Approximations by superpositions of a sigmoidal functions: Math. of Control, Signals, and Systems, **2**, 303-314.

Fahlman, S., 1989, Fast-learning variations on back-propagation: An empirical study, *in* Touretzky, D., Hinton, G., and Sejnowski, T., Eds., Proceedings of the 1988 Connectionist Models Summer School (Pittsburgh, 1988): Morgan-Kaufmann, 38-51.

Fahlman, S., and Lebiere, C., 1990, The cascade-correlation learning architecture, *in* Touretzky, D., Ed., Advances In Neural Information Processing Systems **1**: Morgan-Kaufmann, 524-532.

Geman, S., Bienenstock, E., and Doursat, R., 1992, Neural networks and the bias/variance dilemma: Neural Computation, **4**, 1-58.

Hassibi, B., and Stork, D., 1993, Second order derivatives for network pruning: optimal brain surgeon, *in* Hanson, S., Cowan, J., and Giles, C., Eds., Advances in Neural Information Processing Systems, **5**: Morgan-Kaufmann, 164-171.

Hecht-Nielsen, R., 1990, Neurocomputing: Addison-Wesley.

Hertz, J., Krogh, A., and Palmer, R., 1991, Introduction to the Theory of Neural Computation: Addison-Wesley.

Hornik, K., Stinchcombe, M., and White, H., 1989, Multilayer feedforward neural networks are universal approximators, Neural Networks, **2**, 359-366.

Jacobs, M., 1988, Increased rates of convergence through learning rate adaptation: Neural Networks, **1**, 295-307.

Lapedes, A., and Farber, R., 1988, How neural networks work, *in* Anderson, D., Ed., Neural Information Processing Systems (Denver, 1987): American Institute of Physics, 442-456.

Le Cun, Y., Denker, J., and Solla, S., 1990, Optimal brain damage, *in* Touretzky, D., Ed., Advances in Neural Information Processing Systems, **2**: Morgan-Kaufmann, 598-605.

Masters, T., 1993, Practical Neural Network Recipes in C++: Academic Press.

Parker, D., 1985, Learning-logic: Technical Report TR-47, Center for Computational Research in Economics and Management Science, MIT, April.

Poggio, T., and Girosi, F., 1990, Regularization algorithms for learning that are equivalent to multilayer networks: Science, **247**, 978-982.

Rumelhart, D., and McClelland, J., 1986, Parallel Distributed Processing: Explorations in the Microstructure of Cognition: MIT Press.

Skapura, D., 1996, Building Neural Networks: Addison-Wesley.

Solla, S., Levin, E., and Fleisher, M., 1988, Accelerated learning in layered neural networks: Complex Systems, **2**, 625-640.

Swingler, K., 1996, Applying Neural Networks: A Practical Guide: Academic Press.

Werbos, P., 1974, Beyond regression: New tools for prediction and analysis in the behavioral sciences: Ph.D. Dissertation, Applied Math, Harvard University, Cambridge, MA.

Widrow, B., and Hoff, M., 1960, Adaptive switching circuits: IRE WESCON Convention Record, 96-104.

Wittner, B., and Denker, J., 1988, Strategies for teaching layered networks classification tasks, *in* Anderson, D., Ed., Neural Information Processing Systems (Denver, 1987): American Institute of Physics, 850-859.

This Page Intentionally Left Blank

Chapter 4
Design of Training and Testing Sets

Mary M. Poulton

1. INTRODUCTION

The goal of neural network training is to produce from a limited training set a mapping function or decision boundary that is applicable to data upon which the network was not trained. For the case of continuous-valued outputs we want the network to serve as an interpolator. For discrete-valued outputs we want the network to serve as a classifier. The connection weights are frozen periodically during training so that test data can be applied. Training stops when the error on the test data fails to improve. Only when the test error is as small as possible are validation data applied. How well the net performs on the validation data determines how good the net design is. Both the test and validation data should adequately bound the types of data likely to be encountered in the industrial application of the net. An optimum net design is meaningless if the training, testing, and validation data do not adequately characterize the problem being solved.

Two fundamental questions have to be answered before designing a net: "How do I represent my input and output data and how many training and testing exemplars do I need?" The second question often cannot be completely answered until the first question is settled. Ideally, you should have more training samples than connection weights in the network. Hence, the larger the input vector, the more connection weights and training samples required.

I usually recommend a simple heuristic for the number of training samples of approximately 10 times the number of weights in the network. Baum and Haussler (1989) quantified this for linear nodes and showed that for a desired accuracy level,

$$\text{Accuracy level} = (1\text{-e}), \tag{4.1}$$

where e is the desired error level of the network, the number of training samples can be calculated as,

$$\text{Number of samples} = \frac{w}{e}, \tag{4.2}$$

where w is the number of connection weights in the network. So, for a desired accuracy of 90%, e=0.1, and we need 10 times as many examples as weights.

The network design is an important component of a successful neural network application but the way in which the input and output data are pre- and post-processed and the method for selecting the training data are by far more critical in determining if the application will be successful. Reduction in the size of the input vectors can have a significant effect on the accuracy of the network. If the size reduction, however, comes at the expense of too much information loss, then you will see your accuracy reduced instead of improved. So, much of the success of your network application will hinge on your understanding of the data and how to represent it to the network in the most compact, yet information rich, format. Whenever someone asks me whether a particular problem is suitable for a neural network, my first response is to sketch out what the input and output training patterns would look like.

Besides the preservation of information, the other important constraint on pre- and post-processing data is the computational overhead. In many applications, neural networks provide an extremely fast way to process data (see Chapter 9, for example). If the pre- and post-processing are too computationally intensive, then much of the time savings is lost. When the network application processes data on-line as data are acquired, the time it takes to process the input and output patterns becomes especially critical.

2. RE-SCALING

Every network requires one basic pre-processing step, a re-scaling to a range of [0,1] or [-1,1]. The threshold functions used in most networks require the input to the function to fall within a narrow range of values. The logistic and tanh functions become flat at large and small values and are not sensitive to changes in input values at the tails of the functions. The input scaling is done independently for each input PE. Sometimes, the minimum and maximum values over the entire training set for each input PE in the network are found and that range is mapped to a [0,1] range for the sigmoid function or [-1,1] range for the tanh function. A typical set of linear scaling equations are:

$$m = \frac{network_{\max} - network_{\min}}{data_{\max} - data_{\min}},$$

$$b = \frac{data_{\max} * network_{\min} - data_{\min} * network_{\max}}{data_{\max} - data_{\min}},$$

$$x_i^{scaled} = mx_i + b. \tag{4.3}$$

The variables $data_{max}$ and $data_{min}$ represent the range of values in the training set for each PE since each PE is scaled separately. The variables $network_{max}$ and $network_{min}$ represent the desired range of values for the network calculations, usually [0,1] or [-1,1]. The values in Figure 3.2 that are used by the network after linear scaling are shown in Table 4.1. The output values in Figure 3.2 do not need to be scaled since they are already in the range [0,1].

Table 4.1
Input values from the sample data set in Figure 3.2 after applying the scaling equation (4.3) for a network range of [0,1]

Original data		Scaled data	
x	y	x	y
1	3	0.5	0.5
4	-1	0.7	0.2
3	-5	0.2	-0.3
7	-1	0.9	0.2
8	2	1.0	0.4
-6	12	0.0	1.0
3	4	0.6	0.5
8	12	1.0	1.0
-6	-5	0.0	0.0

If the input data have a normal distribution and do not have extreme values in the training set we can compute a Z-score by calculating the mean and standard deviation for each PE across the training set and then compute the new input to the network as

$$z_{ip} = \frac{x_{ip} - \bar{x}_i}{\sigma_i}, \tag{4.4}$$

and then linearly transforming the z value into the correct range with equation (4.3).

With electrical and electromagnetic geophysical techniques, we may deal with several orders of magnitude for conductivity or resistivity data and therefore need to perform a logarithmic scaling prior to using the linear scaling in equation (4.3). Output data often have to be scaled for training and then scaled back to the "real-world" units before analyzing the results. Frequently, you will find a magnification of error when you do this, especially when a logarithmic scaling is used for training. The network is trained to minimize the error in the scaled data space so when the data are scaled back after training, even small errors from training can become significant. Masters (1993) contains one of the best discussions of data transformation in the neural network literature.

The values of the input pattern may differ considerably in magnitude depending on what measurements they are representing. For example, one input PE may represent seismic amplitude at a point and another input PE might represent rock porosity at that point. The two PEs would have very different values because they are measuring different phenomena. Without rescaling, the network might be more sensitive to the PE with the larger value even though its magnitude is related to its unit of measurement and not its relative importance.

The other benefit of re-scaling the input data is that by forcing all the inputs to fall in a certain range, the connection weights will also fall into that range and we do not have to be concerned about weight values growing too large. By narrowing the range of values for the weights, we decrease the time it takes to traverse the weight space to find the minimum error.

3. DATA DISTRIBUTION

Understanding the data and its distribution is critical. Neural networks do not need to assume data follow a particular distribution, but they can be sensitive to extreme values or class distributions that are highly lopsided. When a classification network fails to provide satisfactory accuracy for some classes, the first place to look for answers is the data not the network. Look at the proportion of training samples in each class and how much the errant classes overlap. If the network is required to estimate a continuous value and you are not achieving the accuracy you desire, you should look at the sensitivity of the output parameters to changes in the input. Geophysical models can always produce a response from a given earth model but sometimes the response is very slight for a large change in the earth model. Most networks will tend to produce an average output response for these cases. Chapter 17 illustrates a method to provide an uncertainty estimate for a neural network estimate so we can distinguish errors due to inadequate training from those due to equivalence.

Table 4.2 shows an example of four 1D earth models that produce nearly identical electromagnetic ellipticity values for a frequency-domain EM system. Ellipticity is a ratio of the imaginary and real induced magnetic fields in a frequency-domain electromagnetic measurement. The models represent a scenario that would be impossible or extremely difficult to resolve with this EM sounding system -- a thin resistive layer. The network is given 11 ellipticity values each representing a different frequency between 1 kHz and 1 MHz and is required to output the parameters for a three-layer earth model consisting of resistivity for each layer and thickness for the first two layers (third layer is a half-space). The neural network estimates nearly identical model parameters although it does make a very slight differentiation between the magnitudes of the second layer resistivity. The values the network estimates for the resistivities of the second layer are not a true average of the models but skewed toward the model that is most easily resolved, i.e. the least resistive layer. The best approach to handle data such as these in the training set is to acknowledge the limitation of the sensor and not include the unresolved cases in the training set.

4. SIZE REDUCTION

Poulton et al. (1992) looked at the effect of size reduction and feature extraction as two means to improve the accuracy of a neural network application. In this example, frequency-domain electromagnetic data were acquired in a traverse over a buried metallic pipe. The data set was two-dimensional with the horizontal axis representing distance along the traverse and the vertical direction representing depth of sounding at each of 11 frequencies. Gridded images produced from a single traverse over the target consisted of 660 pixels (15 interpolated depth points and 44 sounding locations). The network required an input layer with 660 PEs. This represented a large network, but it is still easily trained on today's personal computers. A network this large, however, required a large training set to balance the number of connection weights with training samples, which was not computationally

Table 4.2
Neural network estimates for layer resistivity and thickness of equivalent earth models using an electromagnetic frequency sounding system

Desired Model Parameters					Estimated Model Parameters				
Layer Resistivity			Layer Thickness		Layer Resistivity			Layer Thickness	
R_1 (Ω m)	R_2 (Ω m)	R_3 (Ω m)	T_1 (m)	T_2 (m)	R_1 (Ω m)	R_2 (Ω m)	R_3 (Ω m)	T_1 (m)	T_2 (m)
75	850	40	4	2	72	223	39	3.0	3.0
75	600	40	4	2	75	217	40	3.2	2.9
75	400	40	4	2	73	210	39	3.1	2.8
75	175	40	4	2	77	181	40	3.7	2.2

feasible. Subsampling every other pixel reduced the size of the image. Figure 4.1 shows a sample ellipticity sounding curve for a layered-earth model. The important features in the sounding curves were the locations and magnitudes of the troughs and peaks (minimum and maximum values). These features changed as a function of the geological model. The troughs and peaks along the profile in the 2D image were manually extracted for each depth point. A two-dimensional fast Fourier transform was also used to extract the magnitude, phase, and fundamental frequencies (K_x and K_y). Using the whole image produced the best overall accuracy for the output parameters of (x,y) target location and target conductivity for a small test set. The network using the FFT parameters as input achieved a reduction in size from 660 input PEs to 4 input PEs and produced estimates that were close in accuracy to those

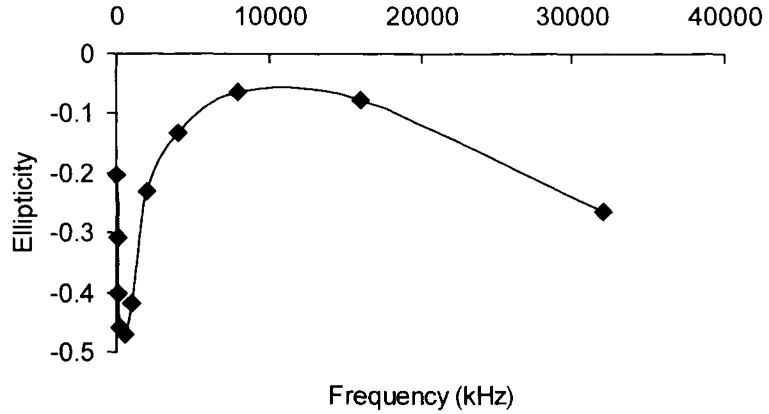


Figure 4.1. Sample ellipticity sounding curve. Important features in the curve are the locations and magnitudes of the minimum and maximum points, or troughs and peaks.

from the whole image. When the differences in accuracy between types of input data representation were compared to differences due to network learning algorithms, the method of pre-processing the data was significantly more important.

5. DATA CODING

Neural networks can work with two fundamental types of data: continuous variables and binary variables. A continuous input variable would be something like seismic amplitude while a binary variable would be a lithologic classification. Binary variables are best presented to the network with a 1-of-n coding scheme. If the output from the network is going to be a lithologic classification of either shale, sandstone, or limestone, then the output coding would be (1,0,0) for shale, (0,1,0) for sandstone, and (0,0,1) for limestone. Most networks will output real values for this output coding scheme so the output from the network for a shale would look something like (0.996, 0.002, 0.073). An additional output scaling using a "softmax" function (Bridle, 1990) can be used to force the output values to 0 or 1 when a 1-of-n coding is used. The softmax function calculates a new output value as,

$$o_{kp}^{new} = \frac{\exp(o_{kp}^{old})}{\sum_{k=1}^{M} \exp(o_{kp}^{old})}. \tag{4.5}$$

Without the softmax function you must decide on a threshold value to apply to the output to determine if the classification is correct. Another alternative, when the tanh function is used, is to code the output as (1,-1,-1), (-1, 1, -1), and (-1, -1, 1) and base the correctness of the classification on the sign rather than the value of the output.

Keep in mind that when you use a binary output coding you should use a non-linear threshold function on the output layer to help force the output PEs to the binary values. The real values that the network calculates have some value in interpreting the accuracy of the output. The closer the values are to 1 or 0, the more confident the network is in the classification and the closer the pattern is to the center of the class. As the output values for the class drift to lower values (i.e. 0.748, 0.325, 0.261), the less confident the classification and the closer the input pattern lies to other class boundaries.

Figure 4.2 shows an arbitrary set of data divided into five classes. A back-propagation network was used to classify the 50 training samples. All of the data points were correctly classified with output values between 0.90 and 1.00 indicating a high confidence in the decision boundaries. The data shown in figure 4.3 were then used to test the network. The network produced values closer to 1.0 whenever a data point was well within the training class and lower values as the data point approached a class boundary.

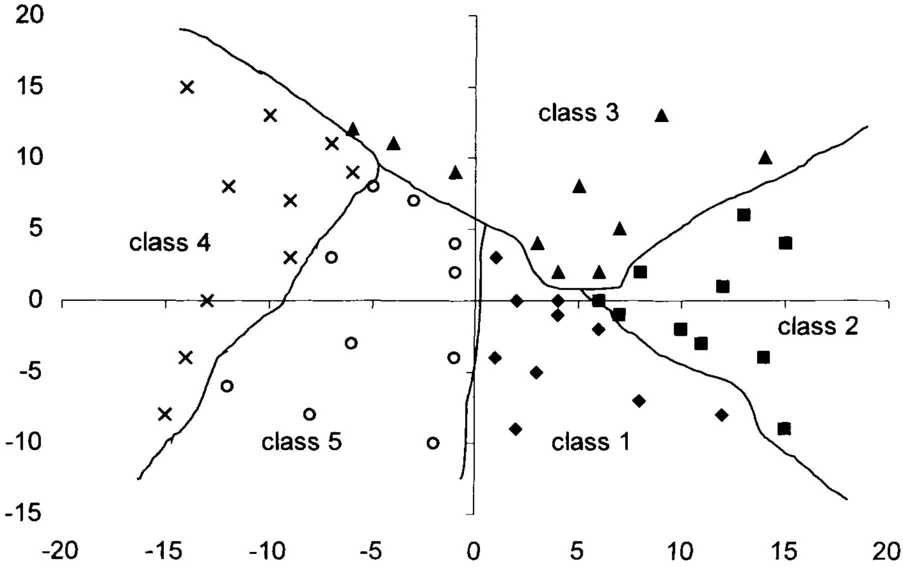


Figure 4.2. Random training data for a classification problem with five output classes.

6. ORDER OF DATA

Computational neural networks do not process patterns the same way the interpreter sees them. A common mistake when working with networks for pattern recognition is to assume that the network uses the same geometric relationships that the interpreter sees in the data to separate the patterns. The network, however, assumes that each input element is independent of the others and the order of the inputs is irrelevant. The values that are acted upon in the network are the weighted sums of the inputs and connection weights so the order of the inputs is lost as soon as the sum is computed. Hence, any important spatial relationships in the data must be explicitly coded in the input pattern or they will be lost.

Electromagnetic ellipticity data were collected at 11 frequencies from 32 kHz to 32 MHz over a simulated disposal pit. Data were classified by a back-propagation network according to whether they represented waste material or background soil conditions. Sounding curves with ellipticity values plotted from low to high frequency are shown in Figure 4.4. This represents the conventional way in which an interpreter would look at the data. The pattern that seems to distinguish waste signatures from background is the low ellipticity values at frequencies 3-5. A neural network trained on 162 patterns such as these easily learns to distinguish between sounding curves from waste material and those from background soil. When the pattern is scrambled and the frequencies are plotted in random order as in Figure 4.5, the pattern is more confusing and almost appears to have the same frequency variations but with different amplitudes. The same network trained on the scrambled data produced exactly the same results as the data presented in frequency order. The frequency for each ellipticity value is not given to the network. The network only works with the ellipticity values.

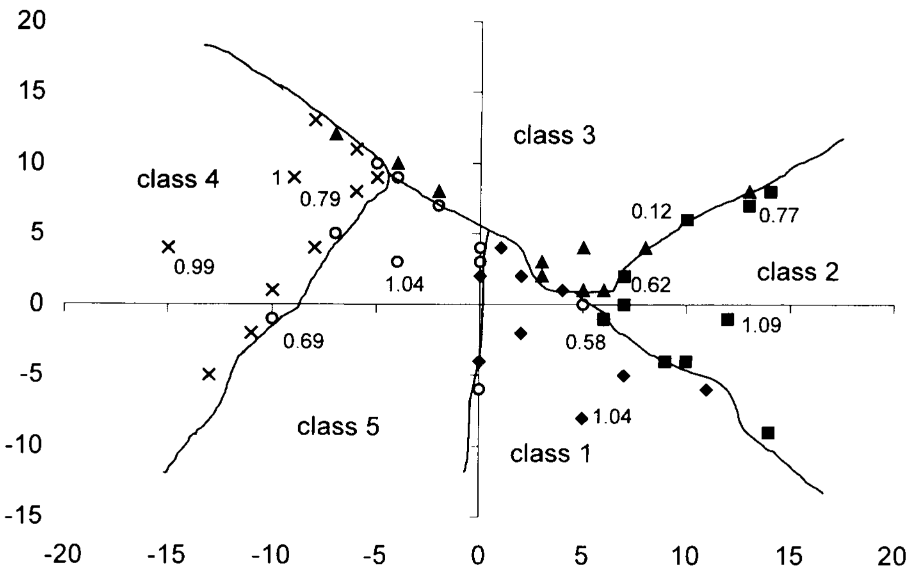


Figure 4.3. Test data associated with the training data in Figure 4.2. Data were chosen to test the locations of the decision boundaries formed during training. Values at selected data points represent the neural network qualitative confidence in the classification of the point. Points well within a class have high values and points near class boundaries have lower values.

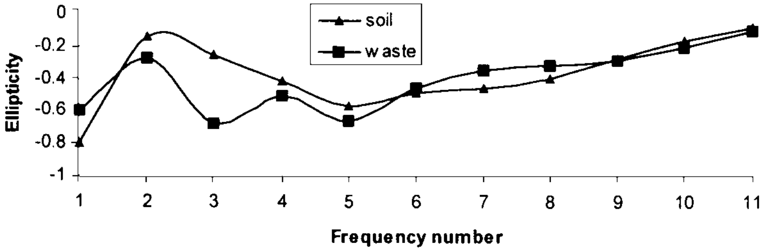


Figure 4.4. Sounding curves for electromagnetic ellipticity values as a function of frequency collected over background soil and buried metallic waste. Data are plotted in a conventional format from low to high frequency (32 kHz to 32 MHz).

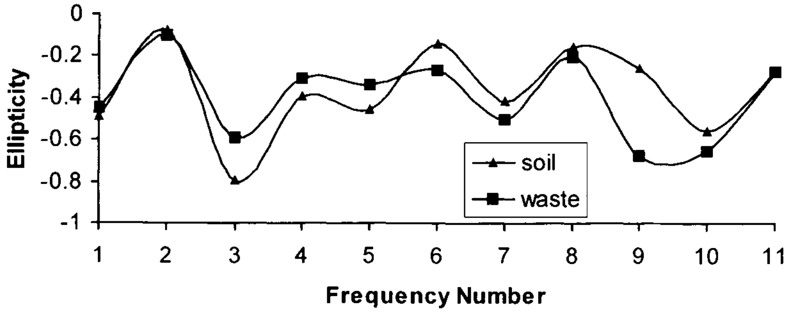


Figure 4.5. The same data as shown in Figure 4.4 but with the frequency values in random order can be learned by a neural network as easily as the patterns in Figure 4.4.

Similar experiments were conducted with input patterns that are classified based on their geometry. For example, a network was trained to classify patterns as letters of the alphabet. The letters were coded in a 5x5 pixel binary matrix shown in Figure 4.6. The network was trained to recognize all 26 letters. Next, the input values were placed in a random order and the same network was re-trained. Again, the network learned all the letters in the same amount of time and to the same degree of accuracy as the network that received the ordered input. Even though the order of the inputs was randomized in the second case, the same random order was used to scramble every training pattern. Hence, the uniqueness of each pattern was preserved but not in the same geometric relation that the eye would use to recognize the pattern. As long as the patterns represent a one-to-one mapping to an output coding (which is also arbitrary), the order of the input values to each PE does not matter. In other words, all we have done by re-arranging the input values is to map one pattern to another but in that mapping we have still preserved the differences between patterns. The important point is that the network does not need to "see" the same pattern the human interpreter sees to solve the problem. Part of the power (and perhaps danger) of neural networks for pattern recognition in geophysical data is that patterns that are too complicated for the human interpreter to reliably use can easily be distinguished by the network. Part of the danger is that the networks may use patterns that are completely irrelevant to the problem at hand. The often-cited example of this is when a network was trained to recognize armored vehicles, even if camouflaged, in photographs. The network easily learned the training data but when tested in the field the network failed. Further analysis revealed that the network was focusing on irrelevant shadow patterns in the photographs and hence had not learned anything about the armored vehicles. Such an example illustrates the importance of a well-constructed training set.

Similarly, a network that must learn a continuous-valued mapping problem also does not care about the order of the input data. A network was trained to map electromagnetic ellipticity sounding curves to layered-earth model parameters of first layer resistivity, first layer thickness, first layer dielectric constant, and half-space resistivity. A network trained with the ellipticity values presented in order from low to high frequency (32 kHz to 32 MHz) produced identical results to a network trained with the ellipticities in random order.

Rumelhart and McClelland (1986) introduced their "Sigma-Pi" units as a way to construct conjuncts of input PEs without having to explicitly code relationships in the input pattern. Instead of simply computing a weighted sum as in equation 3.2, the product of two or more input values is used in place of a single input value. So, equation 3.2 becomes

$$Sum_j = \sum_i w_{ji} \prod_i x_i + w_{jb}. \tag{4.6}$$

The functional link network developed by Yoh-Han Pao in 1989 also addressed the assumption of uncorrelated inputs by introducing a layer of higher-order terms into the network. The functional link layer computes either the outer product of the input pattern with a version of itself (augmented by a value of unity) or some other functional expansion such as a sine or cosine function (Pao, 1989). With the right functional expansion, Pao's networks are able to map the input patterns to a higher-dimensional space that eliminates the need for a hidden layer. The downside to the functional link network is that the expansion becomes enormous for even moderate-sized patterns. Poulton et al. (1992) compared a functional link network to back-propagation and found that once the input pattern exceeded approximately 30 PEs, the functional link network required too much computational overhead to be practical. With the increase in computational power of desktop computers, the upper limit on the functional link network is undoubtedly higher today.

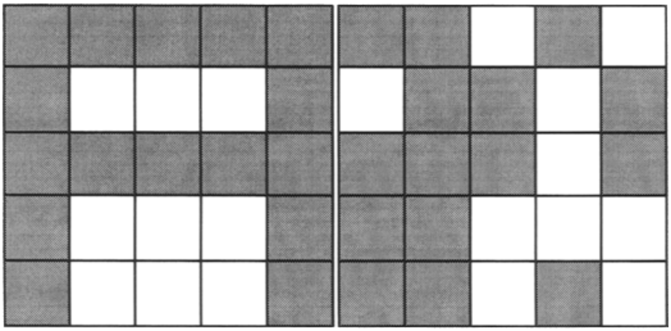


Figure 4.6. A 5x5 pixel matrix showing a geometric pattern representing the letter *A*. In the right-hand figure the order of the pixels is scrambled but the same number of pixels have a value of 1 or 0 as the ordered version of the letter.

REFERENCES

Baum, E., and Haussler, D., 1989, What size net gives valid generalization? *in* Touretzky, D., Ed., Advances in Neural Information Processing Systems **1**: Morgan Kaufmann, 81-90.

Bridle, J., 1990, Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters, *in* Touretzky, D., Ed., Advances in Neural Information Processing Systems **2**: Morgan Kaufmann, 211-217.

Masters, T., 1993, Practical Neural Network Recipes in C++: Academic Press.

Pao, Y., 1989, Adaptive Pattern Recognition and Neural Networks: Addison-Wesley.

Poulton, M., Sternberg, B., and Glass, C., 1992, Location of subsurface targets in geophysical data using neural networks: Geophysics, **57**, 1534-1544.

Rumelhart, D., and McClelland. J., 1986, Parallel Distributed Processing: Explorations in the Microstructure of Cognition: MIT Press.

This Page Intentionally Left Blank

Chapter 5
Alternative Architectures and Learning Rules

Mary M. Poulton

MLP with back-propagation learning can solve a wide range of classification and estimation problems. When the performance of this network is unsatisfactory, either because of speed or accuracy, and we have confirmed that the problems are not due to our data, we have three alternatives: use an alternative to gradient descent; use a hybrid MLP network that maps the inputs to a different dimensional space or partitions the training data into sets with similar characteristics; or use a completely different architecture. We will look at each alternative in this chapter.

1. IMPROVING ON BACK-PROPAGATION

1.1. Delta Bar Delta

It is well known that gradient descent techniques can be slow to converge because there is no guarantee that the steepest gradient is in the direction of the global minimum at any time step. In the back-propagation algorithm, weight updates are always made as a constant proportion (the learning rate) of the partial derivative of the error with respect to the weights so if the error surface is fairly flat the derivative is small and the weight changes are small.

If the error surface is highly convoluted, the derivative can be large which leads to large weight changes. Large weight changes risk the possibility of missing the global minimum. Jacobs (1988) created the Delta Bar Delta (DBD) algorithm to address these three shortcomings of gradient descent. The DBD algorithm uses a strategy of providing a learning rate size for each connection weight. When the derivative of the error with respect to the weights is the same sign for several consecutive time steps, then the learning rate can be increased because the error surface has small curvature in that area. If the derivative changes sign on several consecutive iterations, the learning rate should be decreased because the error surface is highly curved in that area.

In a simpler form DBD increments the learning rate for each connection weight as a proportion (γ) of the products of the partial derivatives of the error with respect to a connection weight at time step (t) and time step (t-1):

$$\Delta\eta_{ji} = \gamma \frac{\partial e(t)}{\partial w_{ji}(t)} \frac{\partial e(t-1)}{\partial w_{ji}(t-1)}. \tag{5.1}$$

If the sign of the derivative is the same for several time steps, the learning rate is increased. If the sign changes, the learning rate is decreased. Unfortunately, if the error surface is relatively flat, the derivatives are small and to compensate γ must be set to a large value. But if the sign of the derivatives stays constant, the learning rate may grow too large over several iterations. If the sign of the derivative alternates and γ is set too large, it will decrease until it becomes a negative number and the weights are adjusted up-slope instead of down-slope. If γ is set too small, convergence is too slow.

The modified DBD algorithm used a different update scheme to avoid these problems:

$$\Delta\eta(t) = \begin{cases} \kappa \text{ if } \bar{\delta}(t-1)\delta(t) > 0 \\ -\phi\eta(t-1) \text{ if } \delta(t-1)\delta(t) < 0 \\ 0 \text{ else.} \\ 0 \end{cases} \tag{5.2}$$

where δ is defined as the partial derivative of the error with respect to the connection weight at time step (t) and $\bar{\delta}$ is defined as

$$\delta(t) = (1-\theta)\delta(t) + \theta\,\bar{\delta}(t-1). \tag{5.3}$$

The variable, θ, is a user-defined value. The $\bar{\delta}$ variable is just an exponential average of the current and past error derivatives with θ as the base and time as the exponent. So, if the error derivatives possess the same sign, the learning rate is incremented by κ and if the error derivatives change sign, the learning rate is decremented by a proportion of the current value.

Jacobs (1988) compared the DBD algorithm to back-propagation for a binary to local mapping problem. The input to the DBD network was a binary string representing a number between zero and seven. The output was an 8-digit pattern that coded the input number according to position in the pattern. For example, if the number 3 was presented to the network the input pattern would be [0 1 1] and the output would be [0 0 1 0 0 0 0 0] (Jacobs, 1988). The back-propagation network needed 58,144 iterations to converge but the DBD algorithm needed only 871 iterations.

1.2. Directed Random Search

Baba (1989) used a random optimization method with a directed component to ensure convergence to the global minimum and speed up the time to convergence. The directed random search (DRS) algorithm includes a self-adjusting variance parameter to increase or decrease learning speed. DRS is a global adaptation technique rather than a local technique more traditionally used by computational neural networks since it makes use of knowledge of the state of the entire network. Baba (1989) found the algorithm to be two orders of magnitude faster than back-propagation for the problems he tested. The basic algorithm is outlined in Table 5.1.

Table 5.1
Algorithm for DRS algorithm

Steps	Comment
1. Select a random weight change $\Delta w(t)$	Draw weight change from a uniform distribution with variance σ
2. Form new weights to test network $w(t+1)=w^*+\Delta w(t)+\Delta c(t)$	(w^* is the best set of weights to date) Δc is a directed component that biases the search in the direction with most past success (use steps 13,14,or 15).
3. Evaluate network for all training samples $e(t+1)=\Sigma(t_{pj}-o_{pj})^2$	
4. $e(t+1)<e^*$?	If true, then continue. If false, then go to step 9.
5. Save new "best" error $e^*=e(t+1)$	
6. Save new "best" weights $w^*=w(t+1)$	
7. Increment counters: $n_s=n_s+1$ $n_f=0$	n_s is the number of successful forward steps n_f is the number of consecutive failures.
8. If($n_s=N_s$) then $\sigma=\sigma^*Ve$ and $n_s=0$	If the number of consecutive successes equals the user-defined limit N_s, then increase the variance by the variance expansion factor Ve. Go to step 13 or 14.
9. $w(t+1)=w^*-\Delta w(t)$	Reversal step used if step 4 was false.
10. $e(t+1)=\Sigma(d_{pj}-o_{pj})^2$	
11. $e(t+1)<e^*$?	If true, then go to step 7. If false, then set $n_s=0$, $n_f=n_f+1$ and continue.
12. If($n_f=N_f$) then $\sigma=\sigma^*Vc$ and $n_f=0$	If the number of consecutive failures equals the user-defined limit N_f, then decrease the variance by the variance contraction factor Vc. The variance is used for the distribution from which the random weight changes are drawn. Go to step 15.
13. $\Delta c(t+1)=0.2^*\Delta c(t)+0.4\Delta c(t)$	Calculate the directed component if step 4 was true. Go to step 1.
14. $\Delta c(t+1)=\Delta c(t)-0.4\Delta c(t)$	Calculate the directed component if step 4 was false. Go to step 1.
15. $\Delta c(t+1)=0.5^*\Delta c(t)$	Calculate the directed component if steps 4 and 11 were false. Go to step 1.

1.3. Resilient Back-propagation

While delivering some improvement over back-propagation, the DBD algorithm still made use of the error derivative to determine the appropriate learning rate for each connection weight. Riedmiller and Braun (1993) have argued that any computation involving the error derivative can lead to poor performance since the derivative may exhibit some unpredicted behavior. The resilient back-propagation algorithm uses the error derivative qualitatively rather than quantitatively. If the error derivative changes sign from one iteration to the next, the last update was too large and the algorithm jumped over a local minimum. Hence, the update value should be reduced by some value η-. If the error derivative retained its sign, then the update value can be increased by η+. If the derivative is positive, the error is increasing and the connection weights are reduced by the update value. If the derivative is negative, the weights are increased by the update value. The algorithm is outlined in Table 5.2.

Table 5.2
RProp algorithm from Riedmiller and Braun (1993)

Steps	Comments
1. If($\frac{\partial e(t-1)}{\partial w_{ji}(t-1)} * \frac{\partial e(t)}{\partial w_{ji}(t)} > 0$)	Check if the sign of the derivative is constant over time
2. Then $\kappa_{ji}(t) = \min(\kappa_{ji}(t-1) * \eta^{+}, \kappa_{\max})$	κ is initially set to an initial value such as 0.1 κ_{max} has an upper limit of 50.0 but 1.0 gives smoother results η^{+} is set to 1.2
3. And $\Delta w_{ji}(t) = -sign(\frac{\partial e(t)}{\partial w_{ji}(t)}) * \kappa_{ji}(t)$	The sign argument returns +1 if the argument is positive and -1 if it is negative
4. And $w_{ji}(t+1) = w_{ji}(t) + \Delta w_{ji}(t)$	If the sign of the derivative in the previous step was negative, then the new connection weights are incremented by κ.
5. If($\frac{\partial e(t-1)}{\partial w_{ji}(t-1)} * \frac{\partial e(t)}{\partial w_{ji}(t)} < 0$)	Check if the sign of the derivative changes over time
6. Then $\kappa_{ji}(t) = \max(\kappa_{ji}(t-1) * \eta^{-}, \kappa_{\min})$	η^{-} is set to 0.5 κ_{min} is set to $1e^{-6}$ $\kappa_{ji}(t)$ will be used in step 10
7. And $w_{ji}(t+1) = w_{ji}(t) - \Delta w_{ji}(t-1)$	Whenever the sign of the derivative changes over time we revert back to a previous weight by subtracting the last change we made.
8. Set $\frac{\partial e(t-1)}{\partial w_{ji}(t-1)} = 0$	To avoid double punishment on the backtracking step, the derivative is set to 0
9. If($\frac{\partial e(t-1)}{\partial w_{ji}(t-1)} * \frac{\partial e(t)}{\partial w_{ji}(t)} = 0$	
10. Then $\Delta w_{ji}(t) = -sign(\frac{\partial e(t)}{\partial w_{ji}(t)}) * \kappa_{ji}(t)$	
11. And $w_{ji}(t+1) = w_{ji}(t) + \Delta w_{ji}(t)$	

The RProp algorithm is compared to back-propagation in Chapter 15 and is found to give superior results in both learning time and test accuracy. The advantage of RProp over the algorithms presented in the next sections is that it can handle large networks with no memory problems.

1.4. Conjugate Gradient

Use of the error gradient to minimize continuous differentiable multivariate functions provides a substantial improvement in efficiency over algorithms that do not make use of the gradient information. The gradient descent technique described in Chapter 3 and used in most standard implementations of the back-propagation learning algorithm is not very efficient for the reasons outlined in Section 1.1 of this chapter. A better approach is steepest descent in which we establish a search direction in the weight space (in the direction of the negative gradient) and then a search step size along that direction. Line search algorithms are a more powerful solution to our problem than gradient descent techniques. The basic line search algorithm involves finding three points, w_i, that bracket the function minimum such that the error $E(w_1) > E(w_2)$ and $E(w_3) > E(w_2)$. Such a relationship implies that the minimum lies between w_2 and w_3. To find the approximate location of the minimum, we can fit a parabola through the three points and use the minimum of that parabola as the approximate location of the minimum of our function. Once we have minimized along a particular line the new gradient will be orthogonal to the previous direction since the direction is always along the negative gradient. This, however, can lead to slow convergence (Bishop, 1995; Hertz et al., 1991).

A better choice than a simple line search is to use conjugate gradients. Once a line has been minimized the next search direction should be chosen such that the component of the gradient parallel to the previous search direction remains zero. If $\vec{d}$ is our search direction, then we want to find a value β such that

$$\vec{d}^{new} = -\nabla e + \vec{d}^{old}\beta. \tag{5.4}$$

The value for β needs to be chosen in such a way that it does not change the minimization already achieved. So we must satisfy the constraint that

$$\vec{d}^{old} \cdot H \cdot \vec{d}^{new} = 0, \tag{5.5}$$

where H is the second derivative Hessian matrix $H_{ij} = \frac{\partial^2 e}{\partial x_i \partial x_j}$. The vectors $\vec{d}^{new}$ and $\vec{d}^{old}$ are said to be conjugate. There are several methods to calculate β: Hestenes-Stiefel; Polak-Ribiere; and Fletcher-Reeves (see Bishop, 1995). Polak-Ribiere is more commonly used of the three and has the form

$$\beta = \frac{(\nabla e^{new} - \nabla e^{old}) \cdot \nabla e^{new}}{(\nabla e^{old})^2}. \tag{5.6}$$

1.5. Quasi-Newton Method

Since the Hessian matrix is not needed in the computation, the conjugate gradient technique can be fast. For a strictly quadratic surface in *n*-dimensional space, the method will converge in *n* iterations (Hertz et al., 1991). Any line minimization technique, however, still involves many evaluations of the error function and these line searches have to be performed very accurately. An alternative to conjugate gradients is Newton's method. Unfortunately Newton's method requires computation of the Hessian matrix and its inverse, which is too computationally expensive for a typical neural network. The quasi-Newton method builds an approximation to the Hessian. Bishop (1995) gives the weight vectors for the quasi-Newton method as

$$\vec{w}(t+1) - \vec{w}(t) = -H^{-1}(\vec{g}(t+1) - \vec{g}(t)), \tag{5.7}$$

where $\vec{g}$ is the gradient vector. We then construct a matrix G that approximates the inverse Hessian. The most used approximation is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) equation (Bishop, 1995)

$$G(t+1) = G(t) + \frac{\vec{p}\vec{p}^T}{\vec{p}^T\vec{v}} - \frac{(G(t)\vec{v})\vec{v}^T G(t)}{\vec{v}^T G(t)\vec{v}} + (\vec{v}^T G(t)\vec{v})\vec{u}\vec{u}^T, \tag{5.8}$$

where $\vec{p} = \vec{w}(t+1) - \vec{w}(t)$, $\vec{v} = \vec{g}(t+1) - \vec{g}(t)$, and $\vec{u} = \frac{\vec{p}}{\vec{p}^T\vec{v}} - \frac{G(t)\vec{v}}{\vec{v}^T G(t)\vec{v}}$.

The weight update is then given by

$$\vec{w}(t+1) = \vec{w}(t) + \alpha(t)G(t)\vec{g}(t), \tag{5.9}$$

where $\alpha(t)$ is found by line minimization. The quasi-Newton method will still converge in *n* steps but has the advantage that the line minimizations do not have to be as accurate as the conjugate gradients method. The disadvantage is that the quasi-Newton method has larger memory storage requirements.

1.6. Levenberg-Marquardt Method

For small networks the fastest and most accurate training technique is usually Levenberg-Marquardt (LM). The LM algorithm is designed to minimize the sum-of-squares cost function without computing the Hessian matrix

$$\vec{w}(t+1) = \vec{w}(t) - (Z^T Z)^{-1} Z^T(\vec{w}(t)\varepsilon). \tag{5.10}$$

The matrix Z is composed of the partial derivatives of the error with respect to the connection weights for each training pattern. The LM algorithm is a type of model trust region method that seeks to minimize the error only in a small search area where a linear approximation is valid. Bishop (1995) gives the modified error function as

$$E = \frac{1}{2}\left\|\vec{e}(\vec{w}(t)) + Z(\vec{w}(t+1) - \vec{w}(t))\right\|^2 + \lambda\left\|\vec{w}(t+1) - \vec{w}(t)\right\|^2, \tag{5.11}$$

where $\vec{e}$ is the error vector for each pattern.

Minimizing the error with respect to $\vec{w}(t+1)$ results in

$$\vec{w}(t+1) = \vec{w}(t) - (Z^T Z + I\lambda)^{-1} Z^T(\vec{w}(t)\varepsilon). \tag{5.12}$$

If λ (a step size parameter) is large, equation (5.12) becomes Newton's method and if it is small the equation become gradient descent. The Levenberg-Marquardt algorithm is explained in more detail in Chapter 11.

The algorithms described in this section were compared using the MATLAB® Neural Network Toolbox (see Chapter 6 for description) for both speed and accuracy on training data. The training set consisted of 4,707 samples of electromagnetic ellipticity curves generated from a two-layer forward model. The thickness of the first layer was constant at 1 m and the resistivity and dielectric constant of each layer were varied. Given the ellipticity value at each of 11 frequencies between 32 kHz and 32 MHz, the network had to estimate the resistivity and dielectric constant for each layer. Each network has 11 input PEs, 4 output PEs and 20 hidden PEs. A tanh activation function was used for the hidden layer and a linear activation function was used for the output layer. Defaults were taken for all variables so no attempt was made to optimize an individual network. Each network was trained for 400 epochs. The rms training error versus number of iterations are shown in Figure 5.1.

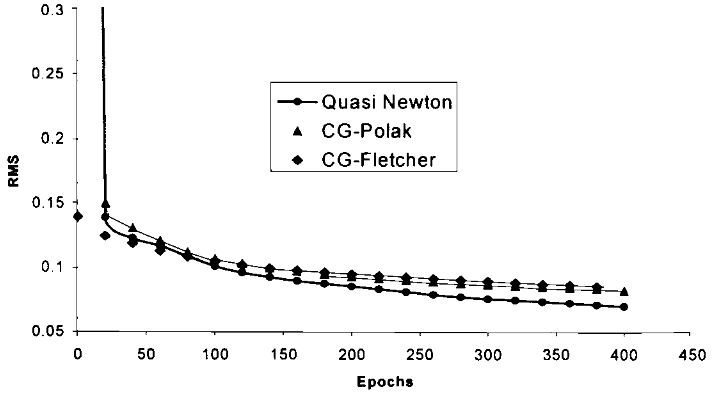


Figure 5.1a. Comparison of rms training error after 400 epochs for a two-layer-earth electromagnetic ellipticity inversion problem. Networks consisted of extended delta-bar-delta (EDBD), MLP with backpropagation learning (BP), and resilient backpropagation (RProp).

Training times for each network are listed in Table 5.3. All trials were run on a 600 MHz Pentium III processor. A Levenberg-Marquardt network was also tried but the training set was so large not enough memory could be allocated. A comparison of the training times and rms accuracies shows the trade-off the interpreter has to make between speed and accuracy. The quasi-Newton technique was able to attain an RMS training error of 0.07 compared to the Rprop RMS of 0.106 but the quasi-Newton network was three times slower.

Table 5.3
Time comparison for MATLAB® networks

Network	Time for 400 epochs (minutes)
MLP-BP	4.75
RProp	4.75
EDBD	4.83
CG-Fletcher-Reeves	10.75
CG-Polak-Ribiere	11.67
Quasi-Newton	12.67

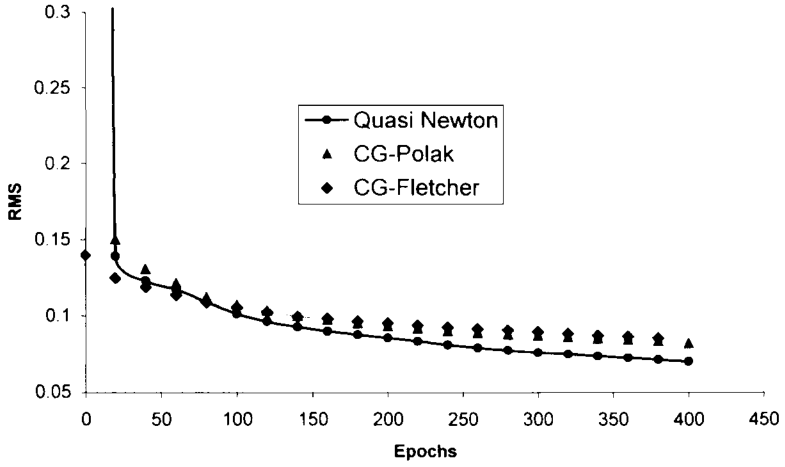


Figure 5.1b. Comparison of a quasi-Newton, conjugate gradient with Polak-Ribiere formula, and conjugate gradient with Fletcher-Reeves formula.

2. HYBRID NETWORKS

2.1. Radial Basis Function Network

Radial basis function networks are described in detail in Chapter 16. The basic premise of this approach is that if we map our input patterns to a higher dimensional space, there is a greater chance that the problem will become linearly separable based on Cover's Theorem

(Cover, 1965; Haykin, 1999). The input pattern is non-linearly mapped to this higher dimensional space through the use of radially symmetric functions (usually Gaussian). Input patterns that are similar will be transformed through the same RBF node. The training process starts with an unsupervised phase during which the center and width of each RBF node must be trained. The centers start with random values and for each input pattern; the center with the minimum distance to the input pattern is updated to move closet to that input pattern. Once the center vectors are fixed, the widths of the RBFs are established based on the root-mean-squared distance to a number of nearest neighbor RBFs. When the unsupervised phase is over, the connection weights between the RBF layer and the output layer are trained with equation 1.7.

The RBF network can also be combined with an MLP and back-propagation learning to produce a hybrid RBF network. The RBF layer is trained unsupervised and its output is used as input to the MLP.

2.2. Modular Neural Network

The modular neural network (MNN) design draws on the structure of the visual cortex in mammals. Nerve cells transmitting signals from the retina are connected to several different networks in the visual cortex at the back of the head. Each network specializes in a different processing step, turning a sequence of dot patterns from the retina into something we could interpret as an image. During early childhood development, nerve cells in the visual cortex compete with each other to see which will respond to signals from each eye. Covering a child's eye for as little as a few weeks can permanently damage their vision in that eye, not because the eye itself is damaged but because the nerve cells in the visual cortex connected to the patched eye cannot compete with the uncovered eye. Once those nerve cells have lost the competition, the biological neural networks cannot establish new connections. In a computational MNN the networks compete with each other for the right to learn different patterns from the training set. Each module or subnetwork learns to contribute a piece of the solution.

Modular neural networks are sometimes referred to as a type of committee machine called a mixture of experts. The network has a basic MLP structure with an input and output layer, but a series of "local expert networks" reside between the layers. These local experts are MLP networks and each is fully connected to the input layer. The output layer of each expert is connected to a "gating network" that determines which of the local experts produced an estimate closest to the desired output. The connection weights in the winning expert network are updated to reinforce the association with that and similar training samples. Hence, the MNN is able to subdivide the training data into regions with similar patterns. The MNN is described in detail in Chapter 15.

2.3. Probabilistic Neural Network

Donald Specht at the Lockheed Corporation published the probabilistic neural network (PNN) in 1990. The idea for the PNN, however, dates back to the 1960s when Specht was a graduate student at Stanford working under Bernard Widrow. The PNN is similar in structure to the back-propagation network but the sigmoidal threshold is replaced by a function derived from Bayesian statistics. The key to implementing a Bayesian approach in a neural network is to accurately estimate the underlying probability density functions from the training data.

Parzen (1962) published an approach to construct such estimates and Cacoullos (1966) extended the approach as,

$$f_A(\bar{x}) = \frac{1}{(2\pi)^{p/2}\sigma^p}\frac{1}{m}\sum_{i=1}^{m}\exp\left[-\frac{(\bar{x}-\bar{x}_{Ai})'(\bar{x}-\bar{x}_{Ai})}{2\sigma^2}\right], \tag{5.13}$$

where m is the total number of training patterns, i is the pattern number, σ is a smoothing parameter, p is the dimensionality of the input space, and $\mathbf{x}_{Ai}$ is the ith training pattern from class A. The smoothing parameter, σ, must be determined empirically. If σ is too small, then each training sample becomes its own class; if σ is too big, too many training samples are grouped into the same class.

Both the input and weight vectors must be normalized to unit length. Equation (5.13) then reduces to

$$o_j = \exp^{(-Sum_j)/\sigma^2}, \tag{5.14}$$

where Sum_j is the weighted sum which is derived from the dot product of the input and weight vectors.

Rather than initializing the weights to random values, the PNN initializes each weight vector to be equal to each one of the training patterns. The PNN then computes the inner product of the input pattern $\bar{x}_p$ and the weight vector $\bar{w}_j$ connecting it to the "pattern" layer,

$$G_j = \bar{x}_p \cdot \bar{w}_j. \tag{5.15}$$

Since $\bar{x}_p$ and $\bar{w}$ are both unit length, equation (5.15) is equivalent to calculating the angle between the two vectors,

$$\cos\theta = \bar{x}_p \cdot \bar{w}_j. \tag{5.16}$$

So, the PNN should output a 1 (cosine of zero) when the input and weight vectors match.

A PNN is shown in Figure 5.2. The network has five input PEs. We have one PE in the pattern layer for each training sample. We have three output classes so we have 3 summation and 3 output units. The PEs in the pattern layer are not fully connected to the summation PEs. Rather, the pattern PEs that have connection weights corresponding to a particular input pattern are connected to the summation unit corresponding to that pattern's output class. So, when a PNN is set up, it is essentially already trained. The only training task to do is activate the pattern and summation PEs as each input is presented. A competitive learning rule is used so only one pattern PE is allowed to be active at a time and each pattern PE is allowed to be active only once. When test data are presented, the input patterns will probably not match any of the existing weight vectors; so, we will calculate a value for equation (5.14) that is something other than 1.0. The summation units will add up all of the values for equation

(5.14) within each class and the class with the largest sum will be the estimated output class for the test pattern.

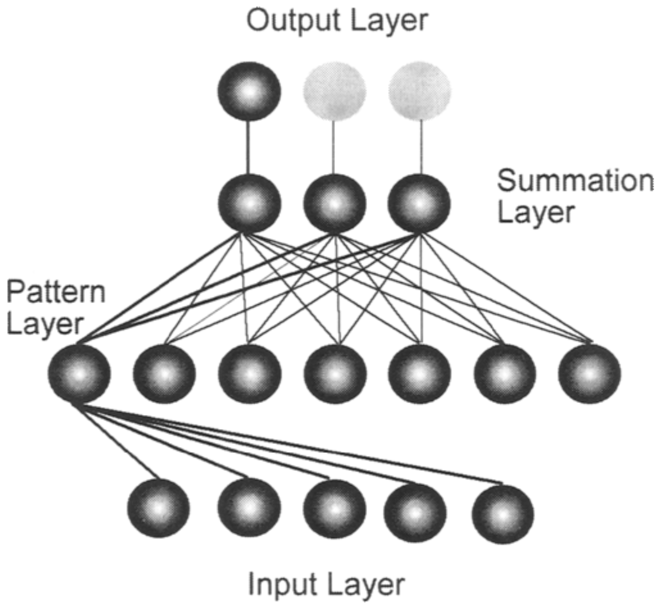


Figure 5.2. A PNN has an input layer fully connected to a hidden layer. The pattern layer has one PE for each training sample. The connection weights between the input and pattern layer are assigned the same values as the input PEs when the network is initialized. One PE in the summation layer will have a maximum response, which will trigger the output classification.

The PNN is primarily useful for supervised classification problems. Unlike the back-propagation algorithm, the PNN trains extremely fast. Unfortunately, it is as much as 60 to 100 times slower than back-propagation when used in recall mode; so, it is not useful in applications where speed is important. The PNN also is not practical for large training sets since it requires one hidden PE for every sample in the training set. The PNN often requires a more comprehensive training set than the MLP but does handle outliers better than MLP (Masters, 1993). Despite these disadvantages, the PNN offers a very significant advantage over the back-propagation algorithm: the ability to calculate the posteriori probability that a test pattern vector $\bar{x}_t$ belongs in class A.

$$P[A \mid X] = \frac{h_A f_A(\bar{x})}{h_A f_A(\bar{x}) + h_B f_B(\bar{x})}, \tag{5.17}$$

provided classes A and B are mutually exclusive and their a priori probabilities, h_A+h_B, sum to 1. Specht (1990) states that the maximum values of $f_A(\bar{x})$ and $f_B(\bar{x})$ measure the density of training samples around the test pattern $\bar{x}_t$ and therefore indicate the reliability of the classification.

2.4. Generalized Regression Neural Network

The Generalized Regression Neural Network (GRNN) was also developed by Specht (1991). The GRNN is a generalization of the PNN but it can perform function estimations as well as classifications. The GRNN also bears similarity to RBF networks. As the name implies, this network starts with linear regression as its basis but extends the regression to avoid assuming a specific functional form (such as linear) for the relationship between the dependent and independent variables. Instead it uses a functional form based on the probability density function (pdf) determined empirically from the training data through the use of Parzen window estimation (Specht, 1991). The result of deriving an estimate of the dependent variable from the pdf is equation 5.18 from Specht (1991)

$$\bar{y}(x) = \frac{\sum_{i=1}^{n} \bar{y}_i \exp(-\frac{z^2{}_i}{2\sigma^2})}{\sum_{i=1}^{n} \exp(-\frac{z^2{}_i}{2\sigma^2})}, \tag{5.18}$$

where $\bar{y}$ is the conditional mean, $\bar{y}_i$ is a training sample, σ is the variance of the pdf, and z_i^2 is defined as

$$z^2{}_i = (\bar{x} - \bar{x}_i)^T (\bar{x} - \bar{x}_i). \tag{5.19}$$

The GRNN usually requires one hidden node for each training sample. To prevent the network from becoming too large, the training data can be clustered so that one node in the network can respond to multiple input patterns. Specht (1991) provides modifications to equation 5.18 for the clustering case.

The GRNN is compared to other algorithms in Chapter 15 and found to be less accurate than RProp, MNN, and BP but no attempt was made to optimize the value for sigma which undoubtedly hurt the performance. Hampson and Todorov (1999), however, achieve excellent results with the GRNN and the algorithm is incorporated into the Hampson-Russell software package EMERGE.

3. ALTERNATIVE ARCHITECTURES

3.1. Self Organizing Map

The self-organizing map (SOM) is based on work by the Finnish scientist, Teuvo Kohonen, who pursued a line of neural network research during the 1970s and 1980s regarding topographical relationships of memories in the brain. The cortex organizes stimuli from different parts of the body in such a way that sensations from your left hand map to an area of the brain very close to the area that receives stimuli from the left arm and so on. Nerve activation in parts of the brain stimulate other nerves within a radius of 50 to 100 μm and inhibit other nerves up to 500 μm away (Kohonen, 1984). The activation pattern of nerves that this creates is often depicted by what is referred to as a "Mexican Hat" function because of its similar appearance to a sombrero (Figure 5.3).

The SOM maps input patterns that are spatially close to each other to PEs that are also spatially close. The preservation of that relationship yields a topographic map of the input. The resulting map shows the natural relationships among patterns that are given to the network. The network has an input layer and a competitive (Kohonen) layer (see figure 5.4); it is trained by unsupervised learning.

We start with an input pattern $\bar{x} = (x_1, x_2, ..., x_n)$. Connections from this input to a single unit in the competitive layer are: $w_{ji} = [w_{j1}, w_{j2}, . . ., w_{jn}]$, where *j* is a PE in the Kohonen layer, and *i* is a PE in the input layer.

The distance between the input pattern vector and the weight vectors for each Kohonen PE is computed using some distance metric (typically Euclidean). The matching value measures the extent to which weights of each unit match the corresponding values of the input pattern.

The matching value for unit *j* is given by

$$Match_j = \sqrt{\sum_{i=1}^{n}(x_i - w_{ji})^2}. \tag{5.18}$$

The unit with the lowest matching value wins the competition. The minimum is taken over all *j* units in the competitive layer. In the case of a tie, we take the unit with the smallest *j* value.

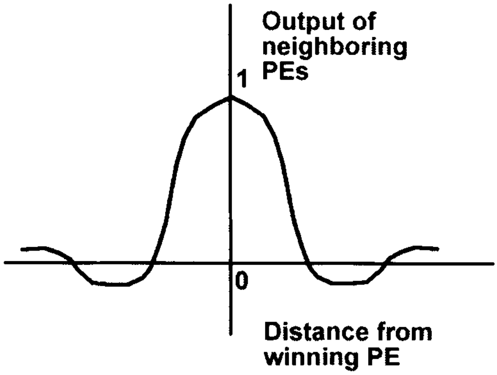


Figure 5.3. The activation of nerve cells in a region of the cortex decreases as a function of distance from the first stimulated cell. In a Kohonen layer, a similar distance function is used within the neighborhood of a winning PE.

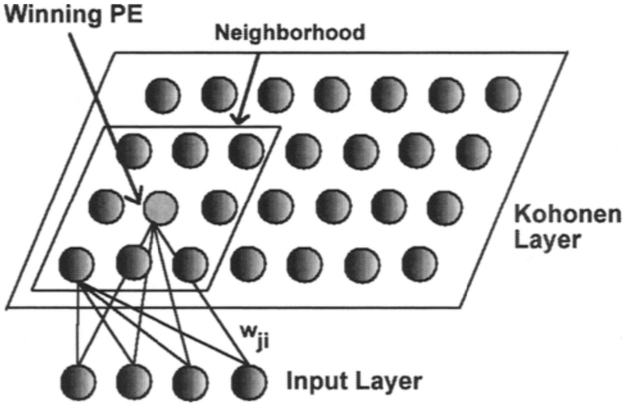


Figure 5.4. The Self-Organizing Map network has an input layer fully connected to the Kohonen layer. PEs within the neighborhood of the winning PE are allowed to update their weights.

The next step is to identify the neighborhood around the winning PE. The size and shape of the neighborhood are variable and are usually on the order of a square with three to five PEs on each side. After the neighborhood is identified, the weights for all PEs in the neighborhood are updated. The Kohonen layer is called a competitive layer because only those "winning" PEs are allowed to make adjustments to their connection weights. The winning PE and its neighbors have their weights modified to become more likely to win the competition should the same or a similar pattern be presented. We calculate the delta weight as,

$$\delta_{ji} = \eta(x_i - w_{ji}), \tag{5.19}$$

if PE *j* is in the winning neighborhood. The delta weights for all other PEs are zero. The weights are updated by adding the delta weight to the old weight for the PE in the winning neighborhood.

Equation (5.19) uses a learning rate or step size parameter, η, which usually starts with a large value and decreases linearly or exponentially during training.

As a simple example, let us construct an SOM to classify some of the data points in Figure 3.1. We can construct a Kohonen layer with 5 PEs, one for each cluster of points. The starting connection weights are listed in Table 5.4. We input the ordered pair (1.0,3.0) from the training set and use equation (5.18) to calculate the distance between the input pattern and weights for each PE in the Kohonen layer. From the fourth column in Table 5.4 we see that the fifth PE in the Kohonen layer had the best match to the input. The connection weights for that PE will be updated according to equation (5.19) and the new values will be 0.79768 and 2.4435, assuming η=0.9. The new weight values are close enough to our input values

(1.0, 3.0) that PE_5 is guaranteed to have the closest match on the next iteration as well, thus reinforcing its association with that particular input value. Assuming we have made the weight change for PE_5, we can apply another input (7,-1) and calculate new matching in column 6 of Table 5.4. Since the ordered pair (7,-1) is in a different class than the previous input pattern, a new PE in the Kohonen layer had the lowest matching value and will be updated.

Table 5.4
Starting connection weights for SOM network and matching values for two input patterns

Hidden Layer	Input PE_1	Input PE_2	Matching Value for (1,3)	Matching value for (7,-1)
PE_1	-0.09976	-0.23166	3.8	7.1
PE_2	-0.59691	-1.37848	4.7	7.6
PE_3	-1.33939	0.23759	3.6	8.4
PE_4	0.30361	-0.29671	3.4	6.7
PE_5	0.11369	0.28433	2.8	7.1

The SOM can operate in an unsupervised mode where the Kohonen layer identifies natural clustering patterns in the data. If coupled with a back-propagation algorithm, the SOM can also operate in a supervised learning mode. In this mode, the Kohonen layer functions as a pre-processor of the input data. The output from the Kohonen layer becomes the input for a back-propagation network that is able to associate a desired output with each given input. The SOM is discussed in more detail in Chapter 10.

3.2. Hopfield networks

In Chapter 1, I attributed much of the resurgence of neural network research to the widely read book on parallel distributed processing by Rumelhart and McClelland (1986). In fact, the legitimacy of neural network research that lead to the explosive growth in the late 1980s can, in part, be attributed to John Hopfield (1984) from California Institute of Technology and a seminal paper he presented in the Proceedings of the National Academy of Sciences in 1982. Hopfield is a highly regarded and very articulate physicist. The neural network model he presented not only was mathematically sound, but it had real, tangible applications in the computer chip industry.

The Hopfield network is different from the other networks we have discussed so far because its function is to be an associative memory rather to use association to perform a classification or estimation. The Hopfield network is also a recurrent network so there is a feedback mechanism between the input and output layers.

The easiest way to understand this architecture is to start with a picture (see figure 5.5). In its most basic form the Hopfield network requires binary input vectors with values of 0 or 1. The network is heteroassociative so the input vector is identical to the output pattern vector and the goal of training is to find a set of connection weights that will reproduce a stored memory when a partial or corrupted memory is presented.

Rather than minimizing an error function, the Hopfield network is developed in terms of minimizing an energy function, E. The connection weights in the network form a matrix W that is symmetrical and has zeroes on the diagonal. To function as an associative memory, the Hopfield network stores patterns in the weight matrix as,

$$w_{ji} = \sum_{p} x_{ip} x_{jp}. \tag{5.20}$$

The diagonal terms, w_{ii}, are set to 0 and the weight matrix is kept symmetric by setting $w_{ji}=w_{ij}$.

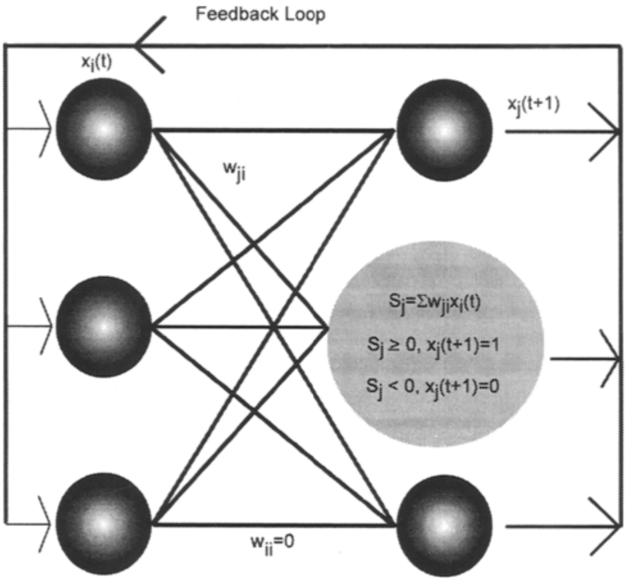


Figure 5.5. Schematic of a Hopfield network. The values of x_i and x_j are either 1 or 0. The diagonal terms of the weight matrix, w_{ii}, are equal to 0.

A weighted sum of the inputs and connection weights is computed as,

$$Sum_j = \sum_{i=1}^{n} x_i w_{ji}, \tag{5.21}$$

and

$$\begin{aligned} Sum_j &\geq 0, x_j = 1 \\ Sum_j &< 0, x_j = 0. \end{aligned} \tag{5.22}$$

The objective function that the Hopfield network minimizes is a Lyapunov energy function defined as,

$$ef = -\frac{1}{2}\sum_j \sum_i w_{ji} x_i x_j \ . \tag{5.23}$$

To see how the energy function must decrease or stay the same, let us look at how the energy is calculated for a specific PE, *j*:

$$ef_j = -\frac{1}{2} x_j \sum_i w_{ji} x_i \ . \tag{5.24}$$

If PE *j* is updated but its value does not change then ef_j does not change. If x_j does change then,

$$\Delta x_j = x_j^{new} - x_j^{old}, \tag{5.25}$$

and

$$\Delta ef_j = -\frac{1}{2} \Delta x_j \sum_i w_{ji} x_i . \tag{5.26}$$

If x_j changed from a value of 0 to 1, then $\Delta x_j = 1$. The weighted sum in equation (5.26) must be greater than 0 so the energy change must be less than 0. If x_j changes from 1 to 0 then $\Delta x_j = -1$. The weighted sum is less than 0 and again the change in energy is less than 0. So, in either case, the energy function decreases.

As a simple example, let us assume that a Hopfield network has three PEs and the connection weights store the pattern {1 0 1}. So, the weight values are those in Table 5.5.

Table 5.5
Weight values required to store the pattern {1 0 1}

	i		
j	0	0	1
	0	0	0
	1	0	0

If we apply the pattern {1 0 1}, we calculate $Sum_1=1$, $Sum_2=0$, and $Sum_3=1$. If we apply a "noisy" version of the pattern {0 0 1}, we calculate the same values for Sum_j and recall our stored pattern. If we apply a different "noisy" version {1 0 0}, we calculate an output pattern of {0 0 1}, which is incorrect. When the pattern {0 0 1}, however, is fed back as input we again calculate the correct pattern {1 0 1}, and no matter how many times this pattern is fed back, the network output produces the same desired pattern; so, it has stabilized.

Wang and Mendel (1992) used a Hopfield network for seismic deconvolution and wavelet extraction. Calderon-Macias et al. (1997) used a Hopfield network for seismic deconvolution and to remove free-surface multiples from seismic data. Both of these applications are summarized in Chapter 8.

3.3. Adaptive Resonance Theory

Adaptive Resonance Theory (ART) was developed by Stephen Grossberg at Boston University in 1980. Human brains are constantly flooded with stimuli and some of that stimulation needs to be stored in memory. New memories can be stored without modifying old memories. This feature of the brain is often referred to as plasticity since it reflects the brain's flexibility in modifying internal structures (long term memory often involves structural changes in the brain). Most computational neural networks, however, do not have the ability to continuously add new patterns to the training set without disrupting the previously set connection weights. The name of the network comes from the similarity between the physical phenomenon of resonance and the way information in the network reverberates between the layers. In its simplest form (ART1) the network works with binary-valued data and can be used as a type of classifier called a novelty detector. ART is capable of recognizing when an input pattern does not fall into any of the existing classes (within some specified tolerance) and will create a new class for the pattern. Hence, it can identify novel inputs without forcing them into an existing class.

The structure of the network is shown in Figure 5.6. The input vector $\vec{x}$ is passed to a comparison layer called the F1 layer. The recognition layer consists of categories (output PEs) and is called the F2 layer. Two sets of weights connect the layers: bottom-up weights from F1 to F2; and top-down weights from F2 to F1. We compute the weighted sum between the input pattern and the bottom-up weights. The F2 layer uses a winner-take-all strategy so only the PE with the largest weighted sum is allowed to remain active. With this approach the network cannot guarantee that the winning PE will respond only to the input pattern that most recently triggered it. In other words, on one pass the PE might represent class {010} and on another pass it might represent class {100}. To prevent this we use the top-down weights for what Grossberg calls "attentional priming". Activity in the active F2 PE should reinforce high activities in the F1 PEs. The exchange of top-down and bottom-up information leads to resonance and critical features in F1 are reinforced (Wasserman, 1989).

The attentional gain control system consists of Gain1 and Gain2, and is used to prevent the top-down signal from F2 from triggering activity in F1 if no input pattern is present (i.e. it prevents the network from hallucinating). The attentional gain control allows the network to distinguish between top-down and bottom-up signals. If input is present, the gain is set to a high value and if no input is present, the gain is set to a low value. Each PE in the F1 layer receives three binary inputs: a component of the input pattern; a feedback signal P_j; and input from Gain1. Two of the three components must be 1 for an F1 PE to remain active. Gain1 is 1 if any component of $\bar{x}$ is 1 but 0 if any component of $\bar{r}$ is 1. Gain2 is 1 if the input vector $\bar{x}$ has any component that is 1.

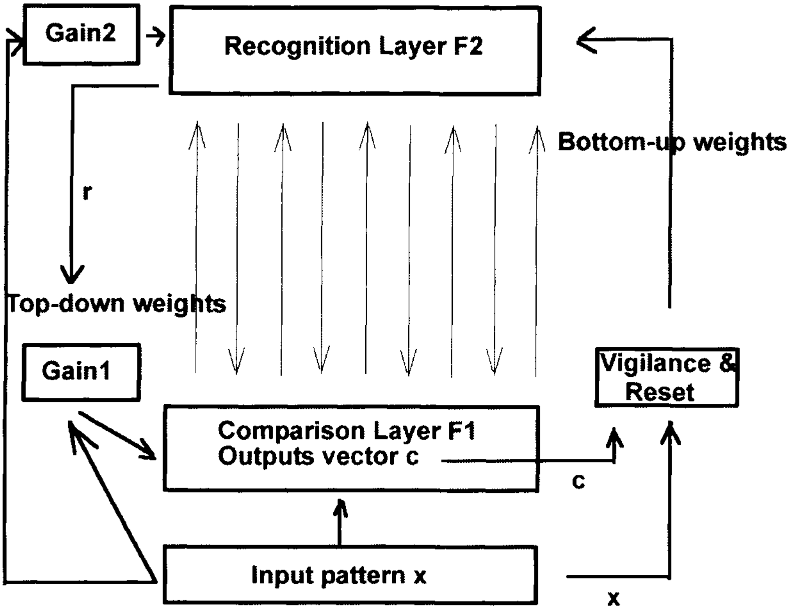


Figure 5.6. Diagram of the ART1 components.

The orienting subsystem, Reset, controls how narrow the class boundaries will be. The orienting subsystem has two inputs and one output. The inputs consist of the input data and the overall activity in F1. The orienting subsystem is connected to F2 and sends a reset wave to F1 whenever the activity in F1 differs from the activity in the top-down weights. The orienting subsystem contains a vigilance parameter that specifies the tolerance within which a pattern must be similar to an existing pattern. If the vigilance limit is exceeded then a new class is created. The flow of information through the ART1 is outlined in Table 5.6.

The ART2 algorithm uses real numbers for inputs and outputs rather than binary values and is described in detail in Skapura (1996). An additional ART algorithm makes use of fuzzy number theory and is called FuzzyART Map. This algorithm allows for fuzzy membership function values to replace the binary values in ART1 so an input in ART1 that was {0, 1, 1, 0} could become {0.2, 0.9, 0.8, 0.1} for FuzzyART Map.

Table 5.6
ART1 Algorithm

Steps	Comments
1. Initialize bottom-up weights $b_{ji} \le \frac{L}{(L-1+m)}$	*L* is a constant, usually set to 2. *m* is the number of elements of the input vector.
2. Initialize top-down weights by setting $t_{ji}=1$ for all i,j	
3. Set vigilance parameter v to value in [0,1]	If *v* is set high, the network will make fine distinctions between classes. If *v* is set low, dissimilar patterns may be grouped into the same class. *v* may be changed over time.
4. Initial state of network is no input vector, x=0, G2=0, all F2 PEs=0	
5. Apply input vector x	*x* must have one or more non-zero elements
6. G1 and G2 = 1	
7. c from the comparison layer F1 is set to exact copy of x	
8. Compute the weighted sum c and b for each PE in F2	
9. Set winning PE, r_j, in F2 to 1.	All other components of *r* are 0
10. Compute priming signal p by computing weighted sum of r and t	Since only one component of *r* is non-zero, only the top-down weights that are connected to the winning PE will have a non-zero value.
11. r is no longer zero so set G1=0	By the 2/3 rule only the F1 PEs will be those that receive simultaneous input from *x* and *p*
12. If there is a substantial mismatch between x and p, few PEs in F1 will fire and c will contain many zeroes while x contains more 1s, indicating an incorrect class in F2 was chosen and the winning F2 PE should be inhibited	
13. Reset block compares x and c, check s<v?	The similarity s=N/D where N=number of 1s in *x*; *D*=number of 1s in *c*. If *s*<*v* go to step 16. If *s*>*v* go to step 14.
14. Set $b_{ji} = \frac{Lc_i}{(L-1+\sum_k c_k)}$	*j* is the winning PE in F2
15. Set $t_{ji}=c_i$	*j* is the winning PE in F2. Go to step 2.
16. Inhibit F2 PE, set r=0, G1=1	
17. Set x=c	Go to step 8

REFERENCES

Baba, N., 1989, A new approach for finding the global minimum of error function of neural networks: Neural Networks, **2**, 367-373.

Bishop, C., 1995, Neural Networks for Pattern Recognition: Oxford Press.

Cacoullos, T., 1966, Estimation of multivariate density: Annals of the Institute of Statistical Mathematics, **18**, 179-189.

Calderon-Macias, C., Sen, M., and Stoffa, P., 1997, Hopfield neural networks, and mean field annealing for seismic deconvolution and multiple attenuation: Geophysics, **62**, 992-1002.

Cover, T., 1965, Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition: IEEE Transactions on Electronic Computers, **EC-14**, 326-334.

Grossberg, S., 1980, How does the brain build a cognitive code?: Psychological Review, **87**, 1-51.

Hampson, D. and Todorov, T., 1999, AVO lithology prediction using multiple seismic attributes: 69th Annual International Meeting, Society of Exploration Geophysicists, Expanded Abstracts.

Haykin, S., 1999, Neural Networks: A Comprehensive Foundation, 2nd Edition: Prentice Hall.

Hertz, J., Krogh, A., and Palmer, R., 1991, Introduction to the Theory of Neural Computation: Addison Wesley.

Hopfield, J., 1984, Neurons with graded response have collective computational properties like those of two-state neurons: Proc. of National Academy of Sciences, **81**, 3088-3092.

Jacobs, R., 1988, Increased rates of convergence through learning rate adaptation: Neural Networks, **1**, 295-307.

Kohonen, T., 1984, Self-Organization and Associative Memory: Series in Information Sciences **8**, Springer-Verlag.

Parzen, E., 1962, On estimation of probability density function and mode: Annals of Mathematical Statistics, **33**, 1065-1076.

Riedmiller, M., and Braun, H., 1993, A direct adaptive method for faster backpropagation learning: The RPROP algorithm: Proceedings of the IEEE International Conference on Neural Networks, 586-591.

Skapura, D., 1996, Building Neural Networks: Addison Wesley, New York.

Specht, D., 1990, Probabilistic neural networks: Neural Networks, **3**, 109-118.

Specht, D., 1991, A general regression neural network: IEEE Transactions on Neural Networks, **2**, 6, 568-576.

Wang, L., and Mendel, J., 1992, Adaptive minimum prediction error deconvolution and source wavelet estimation using Hopfield neural networks: Geophysics, **57**, 670-679.

Wasserman, P., 1989, Neural Computing: Theory and Practice: Van Nostrand Reinhold.

Chapter 6
Software and Other Resources

Mary M. Poulton

1. INTRODUCTION

I have found that one of the best ways to become familiar with computational neural networks is to buy a commercial package that lets you easily train and test several different paradigms. Once proficient with the strengths and weaknesses and appropriate applications for the different networks you can focus your programming resources on those algorithms that are most appropriate to your application. There are two advantages to this approach: 1) you invest resources in several architectures and learning algorithms; and 2) you can quickly prototype an application.

The commercial software packages that include several architectures and learning algorithms and give the user control over most variables seem to be losing ground to the packages that give the user little or no control over the network. Neural network applications have matured to the point where the value is now in being able to bring an application to market quickly rather than researching types of applications and best network architectures. Companies are also finding that increasingly new architectures and learning algorithms are being patented and the cost of keeping a large suite of network architectures up to date is becoming too expensive. Hence, more software titles are offered with proprietary architectures or learning algorithms and fewer software companies offer packages with a variety of networks.

2. COMMERCIAL SOFTWARE PACKAGES

There are at least 33 different commercially available packages and each has strengths and weaknesses. I have chosen some of the more popular packages to review in this chapter.

AspenTech (www.aspentech.com) was formed in 1981 as a technology transfer of the DOE-funded project on Advanced System for Process Engineering (ASPEN) from the Massachusetts Institute of Technology. Based in Cambridge, Massachusetts, the company focuses on "enterprise optimization" software and services. The company broadened its focus from process control technology, primarily in the chemical, petrochemical, refining, and pharmaceutical industry, to optimization of a wider variety of manufacturing processes, largely through a series of strategic acquisitions since 1995. NeuralWare™ was one of those strategic acquisitions in 1997 to provide the company with neural network technology. From the mid-1980s, NeuralWare Professional II/Plus™ and NeuralSIM™ (formerly NeuralWare

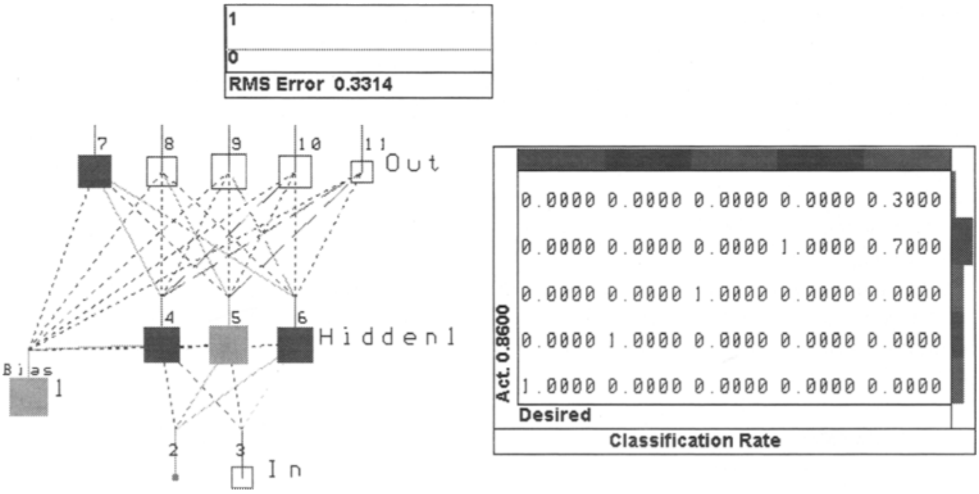
Predict™) have been industry standards in comprehensive neural network packages but after the acquisition they did not have much visibility. NeuralWare is once again independent of AspenTech and will be updating their system.

Professional II/Plus™ contains 22 different architectures or learning algorithms, including MLP, SOM, RBF, modular neural network, PNN, Fuzzy ARTmap, genetic reinforcement, generalized regression, recurrent, and reinforcement networks. The package also includes networks of historical importance such as the Perceptron, ADALINE, bi-directional associative memory, and more. There are two add-on packages, User-Defined Neuro-Dynamics that lets the user change any aspect of a network architecture or learning algorithm, and Designer Pack that allows the user to create source code of the untrained network so it can be trained as part of a custom application.

The interface in Professional II™ (Figure 6.1) shows the PEs and connection weights for the network architecture. PEs are color coded and sized according to their value during training and the connection weights are color coded according to value so the user quickly sees the state of the network qualitatively. Probes can be attached to any part of the network to monitor training status quantitatively. The user specifies the number of input, output and hidden PEs, selects the cost function, activation function, starting values for the learning rate and momentum, whether on-line or batch learning is to be used, whether the training set should be randomized, and how the input data scaling should be performed. The user can also set up a learn/recall schedule that allows the learning rate and momentum values to change over time for each layer. The sensitivity analysis is not as easy to perform as in other packages although it allows the user more flexibility in the way the analysis is performed. The data input and output is from and to an ASCII text file, which is not always as convenient as using Excel. The software is supported on every major platform including SGI. NeuralWare Professional II™ is easy to use with a low learning curve.

NeuralSim™ is designed to optimize the network for the user without the usual trial and error process. The market for this product is the client who needs a neural network application but does not have the expertise in neural network theory required to select all the parameters in the Professional II™ package. The software contains two proprietary training algorithms designed either for clean or noisy data. The program uses an Excel interface for data IO and can create source code for use in C, Fortran, or VisualBasic. The user supplies the data file and Predict selects the data to use for training and testing, choosing the best data transformation, and then creates an optimum network by growing the hidden layer. A sensitivity analysis picks the most relevant variables and uses them only for training. Trained networks can be linked to other applications through a DLL.

NeuroSolutions™ by NeuroDimension, Inc. (www.nd.com) has offered commercial neural network software for sale since 1994. The founding members of the company have ties to the University of Florida and have a medical imaging research background. A demo version of the program can be downloaded free of charge. The demo is easy to install and provides an excellent overview of the package. After spending a few minutes on the demos it is relatively easy to construct, train, and test a basic MLP. NeuroSolutions™ can construct MLP, linear associator, linear adaptive filters, modular, Jordan-Elman, Gamma, time-lagged recurrent, unsupervised, Kohonen, RBF, and principal components (PCA) network architectures. Since the networks are constructed from object classes and families, the interface will appeal to those familiar with object-oriented programming. The software allows users to write macros to customize their applications and also allows OLE and DLL support for communication with other programs. Trained networks can be exported as C/C++ code.



RMS Error 0.3314

```
0.0000 0.0000 0.0000 0.0000 0.3000
0.0000 0.0000 0.0000 1.0000 0.7000
0.0000 0.0000 1.0000 0.0000 0.0000
0.0000 1.0000 0.0000 0.0000 0.0000
1.0000 0.0000 0.0000 0.0000 0.0000
```

Figure 6.1. The user interface in NeuralWare Professional II/Plus™ package. The network is graphically displayed with color-coded connection weights and PEs related to the magnitude of each. The user can display a number of instruments for probes connected to any layer or PE to monitor progress during training (Used with permission of NeuralWare).

The user graphically builds a network on the screen using icons based on objects. A sample screen from the program is shown in Figure 6.2. The two major object families are Axon and Synapse. The Axon represents the processing elements (PEs) and the Synapse represents the connection weights. Selection of FullSynapse means the network will be fully connected while ArbitrarySynapse allows sparse connections. Activation functions are members of the Axon family. The user can select from sigmoid, tanh, softmax, winner-take-all, and Gaussian. The input file used for training is attached to the input Axon icon by dragging a file cabinet icon on top of it. A Control class must be attached to the Axons. The control class specifies the run parameters such as number of epochs for training, how to load and save the weights, and whether to use on-line or batch learning. To monitor the training process a Probe class is used. Probes can be added to monitor the output error, input and output values among other options. After you have connected the Axon and Synapse objects for the number of hidden layers you need, the "forward activation plane" is considered complete. The back-propagation plane (for a MLP) adds a duplicate set of Axons and Synapses for the weight update phase of training. An ErrorCriteria family allows the user to select the standard L2 norm or a L1 or Linf norm for the error calculation. The GradientSearchPlane family adds options of using momentum for training or using the QuickProp or Delta-Bar-Delta modifications.

The networks are fully integrated with Excel; so, the user simply highlights columns in a spreadsheet as belonging to input or output PEs and then tags rows for training and cross validation. A tool called the NeuralWizard simplifies the process by providing step-by-step menus to construct a network application.

In general, the software is powerful and easy to use. The graphical interface and the probes can be overwhelming at first. The Excel interface, however, is very efficient. Sensitivity analysis, which is a vital step in understanding network results, is very straightforward. The choice of architectures is also good and includes some that are not commonly found such as the Gamma and PCA.

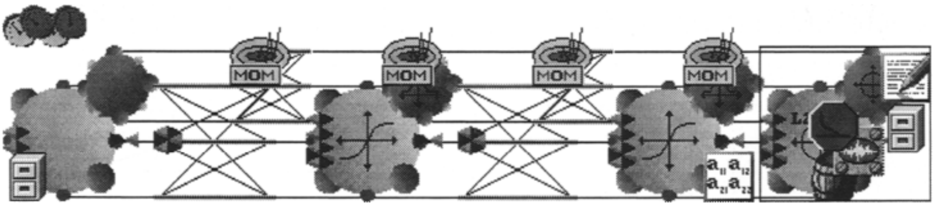
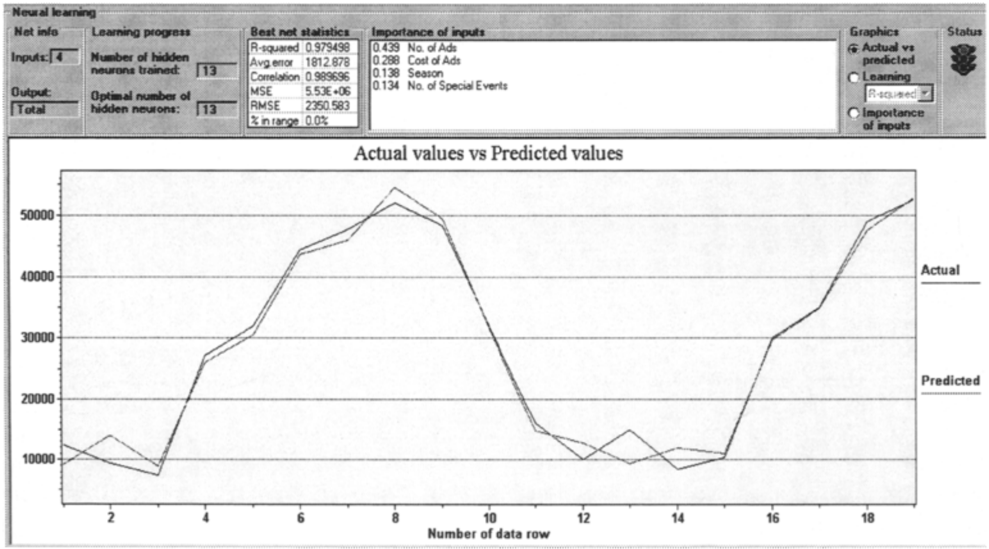


Figure 6.2. GUI for NeuroSolutions™ MLP architecture. The large circular icons are the Axon class for the input, hidden, and output layers. The activation function class is overlain as the sigmoid shaped graph. The Synapse class connects each Axon icon. The smaller circular icons are the back-propagation plane icons. The "MOM" icons are from the GradientSearchPlane class for use of momentum during training. (Used with permission from NeuroDimension.)

Ward Systems Group, Inc. (www.wardsystems.com) first offered a commercial package called NeuroShell™ in 1988. The company now offers a variety of neural network packages customized according to application with a strong bias toward the financial industry. NeuroShell Trader™ is designed for financial applications, such as stocks, bonds, and futures. NeuroShell Predictor™ is the package for regression applications while NeuroShell Classifier™ is the classification package. Neither of these packages allows the user to set parameters. NeuroShell 2™ is a comprehensive package with 16 types of networks that give the user full control over training and testing parameters. GeneHunter™ is the optimization package. NeuroShell 2™ exports C code and also links with Excel. While Ward Systems plans to continue to support NeuroShell 2™, no upgrades are planned and the package does not contain the newest algorithms. NeuroShell 2™ is viewed as more appropriate for the education market than the commercial market. The company, instead, is focusing on commercial applications where the user requires minimal interaction with the network parameters and instead wants to focus on getting a neural network product to market quickly. NeuroShell Predictor™ and Classifier are the tools of choice in this case.

NeuroShell Predictor™ and Classifier™ are designed to be efficient tools for someone who wants to develop neural network applications and does not want to worry about a dozen different parameters or architectures. Predictor uses MLP architecture and grows the hidden layer until an optimum size is reached. The data IO all takes place in Excel where the user highlights the rows of data to be used for training or testing and the columns to be used as input or output. Since there are no architectures or parameters to select, the user only has to choose which graphics to display. The training display is the rms error graph. Figure 6.3 shows the user interface after training that graphs the actual values against the predicted values. Figure 6.4 shows the sensitivity analysis after training. Neural network training does not get any simpler than this. The data files are in Excel for easy manipulation and graphing. Once trained, the network can be saved and applied to any appropriate data files.


Neural learning

Net info
Inputs: 4
Output: Total

Learning progress
Number of hidden neurons trained: 13
Optimal number of hidden neurons: 13

Best net statistics

R-squared	0.979498
Avg.error	1812.878
Correlation	0.989696
MSE	5.53E+06
RMSE	2350.583
% in range	0.0%

Importance of inputs

0.439	No. of Ads
0.288	Cost of Ads
0.138	Season
0.134	No. of Special Events

Graphics
● Actual vs predicted
○ Learning
○ Importance of inputs

Figure 6.3. User interface for NeuroShell Predictor™ showing the fit between actual and predicted values. The software picks optimal network parameters, freeing the user to focus on the data and the application. (From NeuroShell Predictor™ used with permission from Ward Systems Group, Inc.)

California Scientific Software Inc. (www.calsci.com) has marketed a product called Brainmaker™ since 1985. Brainmaker™ uses an MLP architecture with back-propagation learning. A genetic algorithm training option is also available. Like, NeuroShell™, Brainmaker™ has chosen one robust algorithm to take some of the guesswork out of developing an application. The user still has control over several parameters in Brainmaker™. A sensitivity analysis is easy to perform once the network is trained. The user interface is not as sophisticated as other packages, but what it lacks in graphical sophistication it makes up for in speed. The MMX optimized software can train faster than most other packages. A data IO is through a spreadsheet interface and the software is compatible with Excel, dBase, and several financial file formats. Brainmaker™ has a strong user base with over 25,000 copies.

SPSS (www.spss.com) has sold statistical analysis software for mainframes and desktop computers for over 30 years. Recently the company has focused on data mining applications and has become the industry leader in data mining software and consulting services.

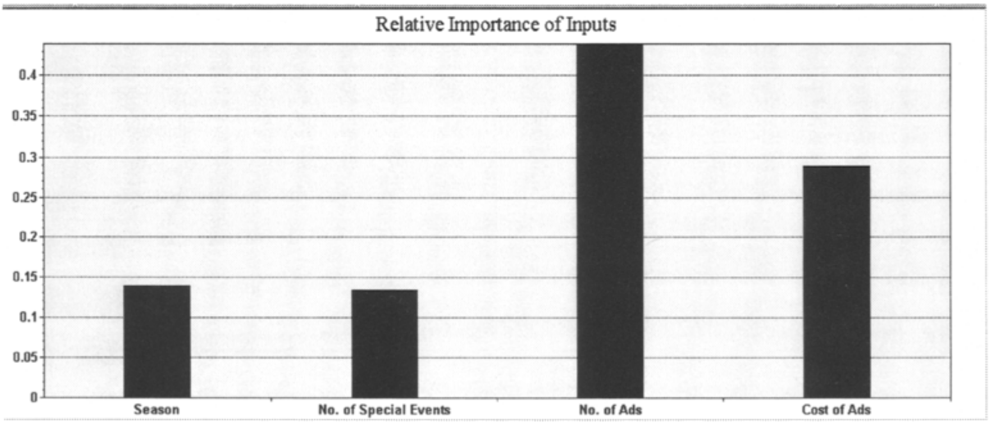


Figure 6.4. A sensitivity analysis of the inputs is available with a single button click in NeuroShell Predictor™. (From NeuroShell Predictor™ used with permission from Ward Systems Group, Inc.)

Emphasis is placed on retail, financial, manufacturing, and marketing sectors. SPSS has developed and acquired a number of products for data mining (SPSS BI®), market research (SPSS MR®), research and engineering statistical data analysis (SYSTAT®), and manufacturing (SPSS Quality®). The neural network software package is Neural Connection® and includes MLP, Kohonen, and RBF networks. Neural Connection® is designed to work closely with other SPSS products for data mining and other applications. The package includes multiple regression, closest mean classification, and principal components analysis to allow easy benchmarking of the network results with traditional statistics. A "What If" tool is incorporated into the program to allow sensitivity analysis of both network parameters and changes to the data. This routine also lets the user quickly generate reports incorporating information about the network parameters and sensitivity. A special scripting language called NetAgent allows the user to write macros that customize the network application. Neural Connection® accepts input from Excel spreadsheets, SYSTAT, ASCII files, and SPSS. The networks allow the user to control all the basic parameters. The MLP architecture with back-propagation is rather basic, offering control over step size, momentum, on-line or batch learning, and sigmoid, tanh, or linear activation functions. As in most neural network packages offered by companies who sell other products, the power and appeal of the software is in the integration with the other products.

StatSoft, Inc., (www.statsoft.com) known for its statistical software package, STASTICA®, has recently released a neural network package called, STATISTICA Neural Networks ®. StatSoft, Inc., based in Tulsa, Oklahoma, was formed in 1984 by a group of university professors who began developing and packaging software for their research data analysis needs. A DOS version of Statistica® was released in 1991 and a Windows version followed in 1993. STATISTICA has an installed user base of 600,000 clients. Like their competitors at SPSS and SAS, StatSoft's growth in recent years has been related to the burgeoning field of data mining. The integration of neural network algorithms with a strong statistical data analysis package allows the user to develop better data warehouse management applications.

The neural network package in Statistica® includes one of the most comprehensive suites of architectures and learning algorithms of any package on the market: MLP with back-propagation, conjugate gradient, Levenberg-Marquardt learning (although the L-M algorithm is limited to a single output PE), quasi-Newton, DBD, and QuickProp; PNN; RBF; Kohonen including LVQ; GRNN; and genetic algorithm optimization. The data interface is designed to integrate with Statistica® but the user can import ASCII and tab-delimited data files. The API included with the package allows the trained network to be fully integrated in other applications. The neural network package has an excellent user interface, allowing the network architecture to be graphically displayed along with a number of analysis tools similar to NeuralWare Professional II/Plus™. One of the strongest features of the Statistica Neural Network® integration is the large number of tools available for exploratory data analysis. Since finding the best data representation and best training set represent the most effort in a neural network application, Statistica® offers a wide range of graphical and statistical tools to accelerate the process of creating a training set.

The SAS Institute (www.sas.com) is the world's largest privately held software company, dating back to 1976. The company has a user base of 3.5 million customers worldwide and is the industry standard software for data mining. Neural networks are integrated into the data mining application called Enterprise Miner®. The neural network tools are limited to an MLP and several variations of the RBF network. Unlike Neural Connection, which can run as a stand alone program, the neural networks in Enterprise Miner® are fully integrated and are intended to run as part of a data mining application.

The MathWorks, Inc. (www.mathworks.com) offers a neural network toolbox with the MATLAB® package. The toolbox contains the MLP architecture and five major variations on back-propagation learning: resilient back-propagation; four types of conjugate gradient; five types of line searches; two types of quasi-Newton algorithms; and two types of Levenberg-Marquardt algorithms. The toolbox also includes subroutines for RBF, generalized regression, SOM, LVQ, and recurrent networks such as the Elman and Hopfield.

MATLAB® is a high-level language for scientific and engineering computing similar to Mathematica® or MathCAD®. MATLAB® stands for matrix library and differs from the other competitors in its use of arrays as the basic data element. Networks are created and trained by linking function calls. MATLAB® and the neural network toolbox have a steeper learning curve than the other packages but also offer more power. What you sacrifice in user interface and easy data IO you gain in control over the algorithms. From a research and development perspective, the toolbox provides strong competition to NeuroSym™ and NeuralWare Professional II/Plus™. If you are interested in getting a product to market quickly with a staff who are not experts in neural networks, then MATLAB® is probably not the best choice.

The following code from MATLAB® is an example of how the user would configure the conjugate gradient algorithm with the Polak-Ribiere formula for the ellipticity-training problem described in Chapter 5.

```
load ep8m10mi.dat;open the file with the input training data, patterns are
in rows in the file
load ep8m10mo.dat; open the file with the output training data
p=ep8m10mi; set variable p equal to the input data matrix
t=ep8m10mo; set variable t equal to output data matrix
for i=1:11
   v(i,1)=min(min(p(:,i)));
   v(i,2)=max(max(p(:,i)));
end; scale the input data
in=P'; create a variable "in" to hold the data from transpose matrix p'
out=T'; create a variable "out" to hold the data from transpose matrix t'
net=newff(v,[20,4],{'tansig','purelin'},'traincgp');set up a new feed
forward network.  The matrix, v, is used for the input data scaling. The
hidden layer will have 20 PEs and the output layer will have 4 PEs.  The
tanh activation function will be used for the hidden layer and a linear
function for the output layer.  Conjugate gradient with the Polak-Ribiere
forumula is specified by the function 'traincgp'.
net.trainParam.show=20; net.trainParam defines the values for the training
parameters.  The show parameter indicates that the rms error will be
plotted every 20 epochs during training.
net.trainParam.lr=0.05; The lr parameter sets the learning rate at 0.05 for
all PEs.
net.trainParam.epochs=400; The epochs parameter sets the maximum number of
training epochs.
net.trainParam.goal=1e-5; The goal parameter sets the training goal to
achieve in order to end training.
net=train(net,in,out); The network is trained with the net object returned
by the newff function call using the variables in and out for the training
input and output data.
```

Partek Pattern Analysis and Recognition Technologies (www.partek.com) package Partek Predict® contains neural network models and conventional statistical algorithms to that allow you to build predictive models.

Z Solutions, LLC (http://www.mindspring.com/~zsol/software.htm) offers another back-propagation package called the BackPack Neural Network System™. The Z Solutions software offers integration of fuzzy logic with the back-propagation algorithm in addition to a proprietary training algorithm.

Attrasoft Software (http://attrasoft.com/) is designed for image recognition, data mining, and financial market prediction. The Attrasoft package uses Hopfield and Boltzmann Machine paradigms.

Thinks™ and ThinksPro™ packages from Logical Designs Consulting (http://www.logicaldesigns.com/) offer capabilities similar to NeuralWare but a bit more limited. Both packages include back-propagation and some of its variants, Kohonen networks, Hopfield networks, Probabilistic networks, and Simulated Annealing.

CorMac Technologies, Inc. (http://www.cormactech.com/neunet/index.html) offers NeuNet Pro™ 2.2. NeuNet is interfaced with Microsoft Access. The software is limited to the back-propagation algorithm and an "SFAM" algorithm that appears to be very similar to the Adaptive Resonance paradigm developed by Stephen Grossberg.

Neural Net Tools™ by Accurate Automation Corporation (http://www.accurate-automation.com/products/nnt.htm) offers C code for nearly every type of architecture. Hard to find architectures include Boltzmann and Cauchy machines, counterpropagation, functional link, adaptive resonance theory and neocognitron. The software can be used to train MIMD hardware under the trademark Neural Net Processor (NNP™). The Multiple Instruction Multiple Data (MIMD) hardware architecture allows up to 10 NNP chips to operate with PCI, ISA, and VME busses. With 10 chips up to 1 billion connections per second can be computed.

3. OPEN SOURCE SOFTWARE

For the more adventurous there is the Learning Benchmark Archive provided by Carnegie Melon University. The Archive (http://www.cs.cmu.edu/afs/cs.cmu.edu/project/ai-repository/ai/html/rep_info/intro.html) contains a range of artificial intelligence applications developed at CMU dating back to 1993.

Stuttgart Neural Network Simulator is made available by the Stuttgart University in Germany (http://www.informatik.uni-stuttgart.de/ipvr/bv/projekte/snns/snns.html). The SNNS runs on most UNIX platforms and supports the most common architectures including MLP with back- propagation learning and its variants(Rprop, backpercolation, recurrent, time delay, Jordan-Elman), Adaptive Resonance (ART1, ART2, and ARTMAP), and Kohonen maps (including LVQ). A recent release of SNNS runs on Windows 95/NT platforms.

Masters (1995,1994,1993) includes in his book C++ source code for several popular architectures and learning algorithms. The MLP architecture supports conjugate gradient training with simulated annealing. Unsupervised training using Kohonen architecture is also included. The software is command line driven and the user supplies values for the relevant training parameters. The code is very well documented. Swingler (1996) also includes C and C++ source code with his book

4. NEWS GROUPS

Other resources include news groups, magazines and journals, professional societies, and books. News groups are a good way to learn about neural networks. Nearly a dozen newsgroups on neural networks can be found, but the most prolific is comp.ai.neural-nets, which is maintained by Warren Sarle of SAS Inc. Dave Touretzky at Carnegie Melon maintains a listserv (connectionists-request@cs.cmu.edu) on neural networks restricted to students and faculty actively researching neural networks. A public listserv called NEURON is available at neuron-request@cattell.psych.upenn.edu.

REFERENCES

Masters, T., 1995, Neural, Novel, and Hybrid Algorithms for Time Series Prediction: Academic Press.

Masters, T., 1994, Signal and Image Processing with Neural Networks: Academic Press.

Masters, T., 1993, Practical Neural Network Recipes in C++: Academic Press.

Swingler, K., 1996, Applying Neural Networks: A Practical Guide: Academic Press.

Part II
Seismic Data Processing

In Part I, the focus was on the fundamental concepts of computational neural networks and we examined some simple examples to gain help understand how the algorithms worked. The following two parts will present different geophysical applications using neural networks. Part II will start with summaries of the applications of neural networks for seismic data processing and attribute analysis. The topics covered are first-break picking, trace editing, multiple removal, deconvolution, and inversion. Chapter 8 reviews work on rock mass and reservoir characterization using neural networks. Classification of attributes represents one of the biggest and most successful applications of neural computing in the petroleum industry and it is examined in more detail in Chapter 10 with a case study.

Neural networks are capable of both classification (supervised and unsupervised) and function estimation. The chapters in Part II will provide examples from each of these categories. Chapter 9 presents a supervised classification example for a marine seismic crew noise problem. The seismic interference present in shot records when more than one crew is working in an area is easily identified by a human interpreter but the vast quantity of data necessitates an automated approach. The neural network application for this problem is presented in detail to help the reader appreciate the importance of each step: 1) pattern identification; 2) input and output data representation; 3) selection of training and testing patterns; 4) network architecture and optimization; 5) analysis of results and assignment of confidence level.

Chapter 10 describes the use of the Self-Organizing Map (SOM) architecture for unsupervised classification of seismic waveforms. The network can group waveforms with similar shapes within a horizon. When those similar waveforms are color-coded and displayed they can be correlated with geological models. Rock Solid Images, Inc., and Flagship Inc. Stratamagic® both use a SOM for waveform classification.

Chapters 11 and 12 present function estimation examples. Chapter 11 compares a MLP with Levenberg-Marquardt learning to standard regression techniques for their abilities to predict permeability from porosity, grain size, and clay content. Chapter 12 develops a method for seismic inversion using a Caianiello neuron instead of the usual McCulloch-Pitts neuron.

This Page Intentionally Left Blank

Chapter 7
Seismic Interpretation and Processing Applications

Meghan S. Miller and Kathy S. Powell

1. INTRODUCTION

Computational neural network applications are promising for automation of some labor-intensive tasks in seismic data processing, such as trace editing, travel-time picking, and velocity analysis as well as applications in seismic interpretation. Among the most promising applications are those that involve seismic attribute processing. A seismic attribute can be broadly defined as any property that is obtained or derived from seismic data. As large 3-D seismic surveys have become more common and computing techniques more powerful and rapid, it has become possible to extract and use more information from seismic data. Hundreds of seismic attributes have been studied, although only 30-50 are commonly used in seismic interpretation (Chen and Sidney, 1997). The greatest amount of research has been conducted on the application of neural networks to analyze and interpolate physical seismic attributes, those that describe lithology, wave propagation, and other physical parameters. This chapter will review neural network applications to waveform recognition, first break identification, velocity estimation, trace editing, deconvolution, multiple removal, and inversion.

2. WAVEFORM RECOGNITION

Palaz and Weger (1990) studied the application of massively parallel networks to a waveform recognition problem. The purpose of this experiment was to investigate the possibility of identifying first breaks on seismic traces. The authors assert that this problem could be applied to many practical geophysical problems, such as recognition of specified waveform on synthetic seismic traces. It is generally assumed that the reflected wavelet has the same waveshape as the incident wavelet, so the seismic trace is actually the superposition of these wavelets (Sheriff and Geldart, 1995). This concept is the basis for calculating the assumed distribution of the physical properties in the earth, which results in a synthetic seismogram, one of the more common types of forward modeling (Sheriff and Geldart, 1995). Synthetic seismograms are commonly used in comparison of actual seismic data to identify reflections with particular interfaces for the creation of lithologic maps.

The network was trained to classify two waveforms as matching or not matching. One waveform was used as the template and successive windows of data were input from a trace while the network computed whether the new waveforms matched the template. Synthetic traces were created on a computer. The traces were normalized to a fixed maximum amplitude value by dividing all the values by the maximum amplitude on the trace. A feed forward network using the back-propagation learning algorithm was designed by trial and error to consist of 4 hidden layers, an input layer with 40 nodes, and one output layer (Figure 7.1). The network consisted of two subnetworks: one for the template waveform and one for the matching waveform. The two subnetworks communicated only at the output PE. The template waveform was fed into groups of five input PEs (1-20) and the matching waveform was fed into similar groups of five input PEs (21-40). Each set of five input PEs fanned out to a single PE in a hidden layer with a total of four PEs in each subnetwork. The next hidden layer had two PEs connected only to the previous four hidden PEs. The third hidden layer had two PEs connected only to the previous two and the final hidden layer had one PE. The output PE received its input from the PE in the fourth hidden layer in each subnetwork. An identical architecture was used for the matching subnetwork.

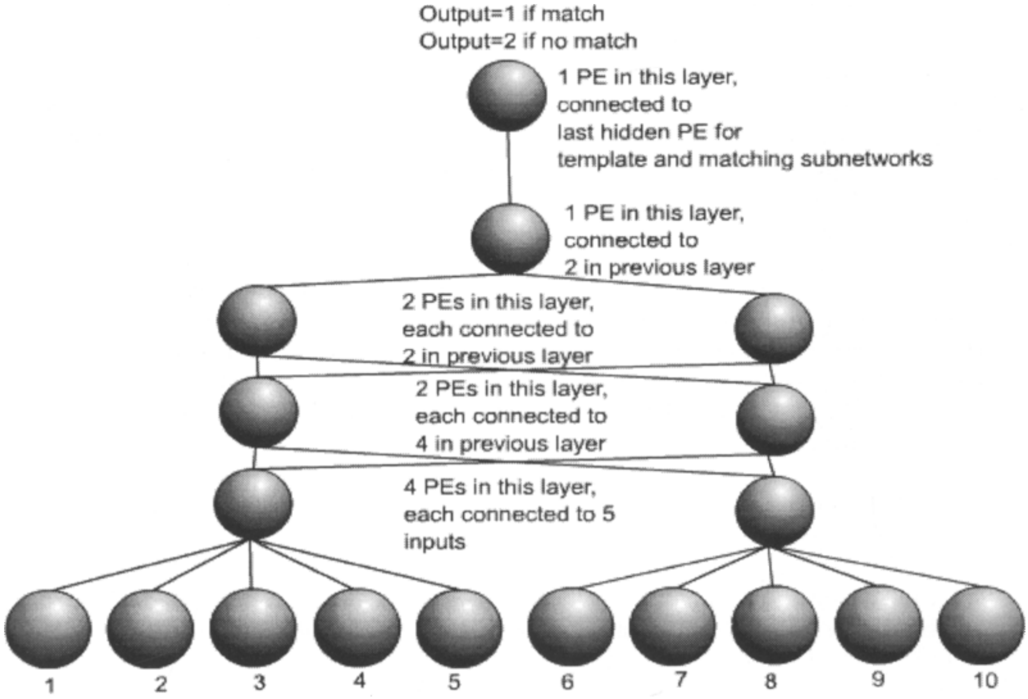

Output=1 if match
Output=2 if no match
1 PE in this layer, connected to last hidden PE for template and matching subnetworks
1 PE in this layer, connected to 2 in previous layer
2 PEs in this layer, each connected to 2 in previous layer
2 PEs in this layer, each connected to 4 in previous layer
4 PEs in this layer, each connected to 5 inputs
1
2
3
4
5
6
7
8
9
10

Figure 7.1. A subset of the network used for waveform matching.

A waveform or trace with 20 sample points at equal time intervals of the amplitude values was selected for the first trial. The 20 sample points were input in the first (1-20) PEs and also the second set of PEs (21-40) and the desired output value was set to "1". The training period continued until the discrepancy between the desired and calculated output was less than 0.001. In testing mode, the same waveform was used for PEs 1-20 and a network tried to match a 20-sample segment input through nodes 21-40. If it failed, then the data were shifted

5 points to look at the next segment from the trace. This process continued until either a match was found or the end of the traced was encountered.

The authors concluded that because a neural network was able to match waveforms, they could potentially be used for first-break picking. It was concluded in this simple experiment that neural networks could accomplish pattern recognition quickly in comparison to other traditional methods. Since the time of the study in 1990, other investigations have concluded that using neural networks to pick first arrivals can be done.

3. PICKING ARRIVAL TIMES

The first breaks are the first arrivals of energy at the geophones. For geophones near the source the first arrival travels along a straight line from the geophone to the source, but with distant geophones the first arrival is a head wave refracted at the base of the low-velocity layer (Sheriff and Geldart, 1995). The arrival times are the interval between the source instant and the arrival of energy at the geophone. The travel time, combined with seismic velocity information, is used to determine the location and attitude of the interface for each reflection event to produce cross-section and contour maps. The maps and cross-sections are used to interpret the geological structure and stratigraphic features. Neural networks are particularly beneficial for this application because of the large volume of data that must be searched to determine the first break arrival times.

McCormack et al. (1993) developed a neural network for first-break picking, which allowed the user to interactively create a new training set and then train a network without having to specify the training parameters. A network-training algorithm is used that adjusts the step size and momentum terms and can prune connection weights without user intervention. By default, a hidden layer is not included in the network unless it fails to converge in which case the user may add a hidden layer with two PEs.

The interpreter creates a training set by simply clicking on first-break picks on a display monitor. A 2D window of data (11 traces by 100 time samples) centered on each first-break pick is extracted from the shot record. The data window is converted from wiggle-trace format to a binary image with 3000 to 4000 pixels. Any pixel that intersects the wiggle-trace is given a value of 1 and all empty pixels are recorded as 0. The binary image is input to the network.

The network learns to classify all positive amplitude peaks on a trace as arriving earlier than the interpreter-identified first arrival or later than the first arrival. All peaks earlier than the first arrival receive an output of (1 0) and all peaks later than the first arrival receive an output of (0 1). The first arrival is then located as the first pixel where the output vector changes from (1 0) to (0 1).

As noted in Chapter 4, when the desired output is a binary-valued pattern, the network will calculate real values and the interpreter must apply a threshold to determine if the classification is correct. McCormack et al. (1993) calculated a reliability factor every time a change in state of the output vector occurred (e.g. change from 1 0 pattern to 0 1 pattern). The reliability factor, r, was computed as

$$r = \left(|o_1 - o_2|^k + |o_2 - o_1|^{k+1} \right) / 2, \tag{7.1}$$

where o_1 and o_2 are the output PEs and k and $k+1$ are the successive peaks at which a state change occurs. If multiple first-break picks are selected on a single trace then the pick with the highest reliability is selected. The reliability factor was also used as a quality control measure for each pick.

The first-break picks were post processed by looking for linear moveout trends on the five adjacent traces left and right of the trace in question. If a pick had a high reliability and fit the linear trend, then it was retained as the true first break.

Veezhinathan and Wagner (1990) attempted to identify the first arrival of seismic energy on a trace by applying a back-propagation neural network to recognize the pattern of seismic first breaks. Peaks in a seismic trace were classified as either first arrival (FAR) or non-FAR. The specific peak to be classified was at the center of a window of five samples and was characterized by a set of signal attributes. In the initial trial, the attributes used were the five peak amplitudes in the window, the mean power level in the window, and the power ratio between a forward sliding and reverse sliding window. The back-propagation neural network used a generalized delta learning rule. The network configuration consisted of seven input neurons, three hidden neurons, and one output neuron. If the output response ≥ 0.8, the peak was considered FAR. The network was trained and tested on data from a marine seismic survey.

Results of the neural network classification were evaluated in two ways – percent of peaks classified correctly and percent of traces absolutely correctly picked. The authors defined a trace as being absolutely correctly classified if the FAR is correct and no false alarms were raised in the trace.

When the network was tested on traces from the adjacent ten records, 92% of the peaks were classified correctly, but only 65% of the traces were absolutely correctly classified. Veezhinathan and Wagner (1990) attributed this poor performance to false alarms of post-FAR peaks and concluded that the network's discrimination ability between FAR and post-FAR peaks needed to improve.

The authors subsequently used a neural network based on four signal attributes. Besides the mean power level in the window and the power ratio between a forward sliding and reverse sliding window, only the maximum peak amplitude was used (instead of all five amplitudes) along with a new attribute, the envelope slope, to characterize a peak. The sample window contained three consecutive peaks to decide if the central peak was a FAR. One peak on either side of the central peak provided a spatial correlation for identifying the first arrivals. This network had a total of 12 input neurons, five hidden neurons, and one output neuron.

The output was post-processed to remove false alarms in individual traces, since the network did not consider individual intertrace correlations.

The new network was trained with 81 traces from a single seismic line of a Vibroseis® survey. It was then tested with 20,000 traces from five different seismic lines, some of which were thousands of feet away from the training seismic line. The network achieved above 95% final accuracy with an improvement in turn-around time of 88%. Testing on data from a marine seismic survey yielded comparable results (Veezhinathan and Wagner, 1990).

This work was extended in later research by Veezhinathan et al. (1991) in which a least-squares line was fit through all the FAR peaks identified by the network for a shot record. The least-squares line allowed the linear trend of the FAR peaks between traces to be taken into account and improved the accuracy of the system. This eliminated some false alarms and permitted FAR estimates for traces which had no FAR peaks identified by the neural network.

To replace the post-processing step in the previous paper, the authors added a fifth attribute, the distance to a reduced travel-time curve, as another input feature to the neural network. The distance to reduced travel-time curve is an indicator of the linear trend of the FAR peaks. This network configuration achieved comparable results to those obtained by using the post-processing step (Veezhinathan et al., 1991).

Murat and Rudman (1992) applied a back-propagation neural network to identify first breaks in seismic data. They first trained and tested the net on data from a simulated seismogram. Input consisted of three consecutive amplitude values of the seismogram within a sliding window. An output of (1) indicated a FAR match, (0) indicated non-FAR.

The network had three input PEs, one hidden layer with ten PEs, and one output node. The generalized delta learning rule was used by the net, along with the sigmoid activation function. The network was trained for 315 iterations and then tested. Although the network did identify the true FAR, it also identified other patterns as FAR with even higher confidence. Results were deemed unsatisfactory and the authors concluded that the network's discrimination ability to identify first breaks needed to be improved.

Potential seismic attributes to be used as input to the network were evaluated using 3-D decision region plots of attributes from 20 traces of a Vibroseis profile. The 3-D plots indicated that four attributes would allow first breaks to be identified uniquely: peak amplitude of a half-cycle, peak-to-lobe difference, RMS amplitude ratio, and RMS amplitude ratio of adjacent traces. The fourth attribute provided a measure of spatial coherence for the first breaks with regard to intertrace correlation.

The new network consisted of four input PEs, one hidden layer with ten PEs, and one output node. It was trained using data from four non-adjacent traces of a real Vibroseis profile. The network then correctly selected first breaks for the remaining 116 traces of the training profile with a confidence over 0.9 for each. First breaks were also correctly identified for a profile obtained at some distance away from the training profile.

Murat and Rudman (1992) tested the network with poorer quality data obtained using a Poulter source (air shooting). The network was trained on Poulter data and then tested with 96 traces. First breaks were selected with 99% accuracy. Data were again obtained from a profile recorded far from the training profile and input to the network. It correctly identified 70% of the first breaks. The authors concluded that this method of first break identification provided accurate results and could be improved by eliminating picks with a low level of confidence or which deviate from a linear intertrace trend.

An and Epping (1993) investigated three different methods of characterizing a potential FAR peak for use as input features to a neural network. The three representations of the candidate peak were created using seismic attributes, peak amplitudes, and the wavelet transform. The first method used four seismic attributes including peak amplitude, envelope slope, mean power, and power ratio. The values of these attributes for the central candidate peak, along with the two peaks adjacent to it, were used as the full feature input vector.

The second method utilized the amplitude values of peaks for a window of trace samples centered about the candidate FAR peak. The number of appropriate trace samples was determined to be about twice the time between the FAR peak and the following peak, or about 19-23 samples for their data. The authors proposed that the poor results obtained in previous research involving peak amplitudes were probably due to the use of too few peaks (generally three to five).

The third technique of representing a candidate FAR peak, the wavelet transform method, included both time and frequency information. "The wavelet coefficients are given by

$$W(a,t) = \frac{1}{\sqrt{a}} \int h\left(\frac{t'-t}{a}\right) A(t')dt', \tag{7.2}$$

where $A(t')$ is the seismic trace, and $h(t) = (1-t^2)\exp(-t^2/2)$ is the "mother" wavelet. The wavelet coefficients $W(a,t)$ are evaluated at discretised scale a and time t. The scale a is discretised by $a = a_0^j$, i.e., raising the basic scale a_0 to the power j, and the time t is discretised as $t = nt_0$. Both j and n are integers. In the present study, [the authors] took 5 scales ($j = 0,1,\ldots4$) and N time samples ($n = 1,2,\ldots N$), with the basic scale $a_0 = 1.3$ and the sampling interval $t_0 = 4$ ms. These wavelet coefficients are then used to define the feature vector for the peak at the centre of the N-sample time window" (An and Epping, 1993).

The back-propagation neural networks utilized by the authors had one hidden layer and used the tanh(x) transfer function. The number of neurons in the hidden layer was determined experimentally for each case, while the number of input neurons were set by the dimension of the input feature vector. The networks were trained using manually picked first arrivals from a shot-gather of 150 traces that had been normalized. A peak was classified as a FAR if it had the highest probability for a particular trace and that probability was greater than 0.5.

The neural networks were trained and tested on both dynamite and Vibroseis data. A trace was considered correctly classified if the peak selected by the network as the FAR was the same one chosen by a seismologist. The three different input characterizations yielded similar results for each type of source. The networks correctly classified 88-94% of the dynamite data. The classifications of the lower signal-to-noise Vibroseis data were only 55% accurate.

Dimitropoulos and Boyce (1993b) performed a comparison of neural network algorithms used for first break identification based on accuracy, optimization of network architecture, learning rate, and generalization ability. Their study included networks that used both back-propagation and cascade-correlation algorithms. The comparisons were conducted on a Meiko transputer surface that used both transputers and i860's. Different methods were first analyzed using dynamite reflection data; the optimum results were then applied to Vibroseis data.

Three techniques of characterizing data from a trace for use as input were investigated, based on amplitudes of samples, peak amplitudes, and seismic attributes. A sliding time window was used to extract information from each trace for all the methods. For the amplitude method, the sliding window moved along the trace in steps of one sample at a time. The other two methods only used windows that were centered around peaks in the trace.

The first method used the amplitudes of trace samples within the window, scaled to the range [0, 255], as the input to the neural networks. The number of input PEs was based on the size of the window. For a sampling interval of 4 ms, the dynamite data had 250 samples per trace. The best results were obtained with a window size of 75 samples.

The networks used two output PEs. An output of (0,1) indicated that the first break was located below the current sample; an output of (1,0) indicated the first break was above the current sample. The position of the first break was estimated to be the point at which the output vector changed from (0,1) to (1,0). A first break was considered to be correct if the position selected by the network was within one sample deviation (4 ms) of the actual first break.

The optimal back-propagation neural network had 75 input PEs and one hidden layer of eight PEs. The network was trained on data from 36 traces and validated with 12 traces, using 5000-6000 iterations. Training times ranged from 3-5 hours. For the dynamite data, the amplitude method achieved 94-96% accuracy. About 70% of the first breaks were chosen at the true sample location. This method was considered inappropriate for the Vibroseis data due to excessively large training sets and times.

The optimal cascade-learning neural network consisted of 75 input neurons and 17 hidden single-node layers. The output PEs were changed from two to one, with (0) indicating samples before and (1) indicating samples after the first break. The net was trained using 2300 iterations and lasted 4 hours. The amplitude method attained 96% accuracy, but only 42% of the first breaks were chosen at the true sample location. This method was again considered inappropriate for Vibroseis data. The authors concluded that the results from the back-propagation neural network are more accurate when using the window amplitude method.

The second approach tested by Dimitropoulos and Boyce (1993b) only the amplitudes of peaks within the sample window. The window was centered about a candidate peak that had an amplitude above a certain threshold; otherwise it was considered noise. The net picked the first break at the peak where the output vector changes from (0) to (1).

For the peak amplitude method, the cascade-correlation net achieved far better results than the back-propagation one. The network needed only one hidden neuron and trained in less than 35 seconds to obtain 98% accuracy for the dynamite data. The back-propagation network achieved comparable accuracy, but took 10-40 times longer to train because the optimal number of hidden neurons must be found by trial and error.

The Vibroseis data required a gating window be applied to the trace to limit the number of candidate first break peaks to eight per trace. A window of 75 samples was again used as input features. The optimal cascade-correlation net created 19 hidden neurons and required 4-5 minutes training time. The accuracy of the network was 52%.

The final technique of characterizing signal data involved pre-processing to extract a set of attributes for the samples within each window. The four attributes calculated for each window were peak amplitude, peak-to-trough amplitude difference, root mean square (rms) amplitude ratio, and rms amplitude ratio for adjacent traces. The number of input PEs was thus reduced from 75 to four.

The cascade-correlation network was again considered the superior network. The optimal net had four inputs, seven hidden layers, and 15-second training time. The network achieved 98% accuracy on the dynamite data. For the Vibroseis data, a network of 42 hidden PEs using two minutes of training time, yielded a performance of 72% accuracy.

Dimitropoulos and Boyce (1993a) concluded that the peak amplitude method with the cascade-correlation network was the best approach to use for dynamite data, since it was simple to implement and accurate. For Vibroseis data, the attribute method with the cascade-correlation network was preferred due to its small number of input neurons and short training times.

Chu and Mendel (1994) applied a back-propagation fuzzy logic system (BPFLS) to first arrival identification. Their multi-input, single-output BPFLS can learn from training samples like a neural network, but also from subjective linguistic rules by human experts.

The fuzzy logic system is analogous to a three-layer feedforward network. Non-fuzzy input is put through a fuzzification interface, then a fuzzy inference machine where the fuzzy rule base is applied, and finally a defuzzification interface, from which non-fuzzy data is output. The fuzzy system itself used product inference, Gaussian membership functions, and the height method of defuzzification.

Seismic data were pre-processed to obtain candidate FAR peaks to be input to the fuzzy system. Candidate peaks were chosen based on whether the peak was a local maximum and its value greater than a particular threshold. Five attributes of the candidate peaks were used as inputs of the BPFLS, including the maximum amplitude, mean power level, power ratio, envelope slope, and distance to guiding function. Using the distance to the piecewise linear guiding function provides a method of including intertrace lateral variations into the system. A training sample consists of the five attributes of a three-peak group (total of 15 inputs) and their associated output (FAR or non-FAR).

Chu and Mendel (1994) performed a number of simulations and determined that the number of training samples and rules affected the approximation capabilities of the BPFLS. The distance to the guiding function attribute yielded better results than not using it, and linguistic rules could be substituted for the distance to guiding function with comparable results. An example of a simple linguistic rule: "If the distance between the candidate FBR peak and the guiding function is small, then it is likely to be a FBR peak." The authors also compared the BPFLS to the BPNN described by Veezhinathan et al. (1991) and conclude that the BPFLS achieved similar picking accuracy but with a much faster convergence rate. They proposed that this was due to the systematic method in which the initial parameters of the BPFLS were chosen, compared to the random weights initially used by the neural network (Chu and Mendel, 1994).

Dimitropoulos and Boyce (1994) applied a supervised, but self-organizing Adaptive Resonance Theory (Fuzzy-ARTMAP) neural network to first arrival picking in seismic reflection data (see Chapter 5 for a description of Fuzzy-ARTMAP).

A sliding time window was applied to each trace and 18 input attributes were extracted from the window, using the Peaks-Troughs-Distances-Adjacent RMS (PTDA) method. Each window contained a central candidate peak. The central peak amplitude value, along with the values of the two peaks on either side of it, were used as the five peak attributes. The amplitude values of the four troughs between the peaks and relative distances between peaks and troughs (8 distances) were also used. The rms amplitude ratio was calculated on adjacent traces and allowed the network to take the spatial correlation of FAR picks into account. The two rms amplitude ratios of adjacent traces were added together for this final input attribute. One output neuron was used to indicate whether a peak was before (0) or after (1) the first break. A change in the output from (0) to (1) indicateed the position of the FAR.

The dynamics of the two Fuzzy-ART modules were determined by three parameters: The choice parameter, $\alpha>0$; the learning rate parameter, $\beta\in(0,1)$; and the vigilance parameter, $\rho\in(0,1)$. Only one training iteration is required for each parameter configuration, since the initial weights are set to 1.

Dimitropoulos and Boyce (1993b) tested various parameter configurations on both dynamite and Vibroseis seismic data and compared the best results with those obtained from a cascade-correlation neural network. The Fuzzy-ARTMAP estimates for first breaks were 2-8% less accurate than the cascade-correlation network, but generally required less computer time. The authors concluded that Fuzzy-ARTMAP is a viable candidate for first break identification because of its accuracy, speed, stable learning, ability to *not* get trapped in local minima, and capability to extract fuzzy rules used to map input to output.

4. TRACE EDITING

One of the most labor-intensive areas of seismic data processing is the editing of noisy seismic traces because human expertise is needed to make these subjective decisions. Data with anomalously high amplitudes, probably due to noise, can be reduced to zero or to the surround amplitude (Sheriff and Geldart, 1995). In the past programs have attempted to reduce the time needed to perform this task, but they have been limited by their accuracy and reliability when signal and noise conditions changed during the course of the survey.

McCormack et al. (1993) used a two-layer MLP with back-propagation learning to train a network to classify seismic traces as good or noisy. The interpreter picked examples of good and noisy traces from a data set. The FFT amplitude spectrum was calculated for each trace along with the average trace frequency, average trace energy, average absolute amplitude, ratio of two largest peaks, energy decay rate, normalized offset distance, cross-correlation between trace and two adjacent traces, and average trace energy compared to four adjacent traces. The resulting input pattern had 520 elements, 512 of which were from the FFT. The output was (1 0) if the trace was clean and (0 1) if the trace was dead or noisy. A reliability factor was calculated for each trace as the absolute value of the difference between the two output nodes. The interpreter and the network agreed on their classifications 95% of the time in the trial runs.

5. VELOCITY ANALYSIS

Ashida (1996) trained an MLP to perform velocity analysis in a two-stage process. In the first stage the reflected wave was recognized and in the second stage a velocity spectrum was computed. To detect the reflected wave, Ashida (1996) used a similar approach to that of Palaz and Weger (1990) in Section 2 of this chapter. The MLP had 40 input PEs, 8 PEs in the first hidden layer, 4 in the next two hidden layers, 2 in the fourth hidden layer, and a single output PE. A Ricker wavelet was generated with a peak frequency of 30 Hz and random noise is added. The synthetic seismic data used for training had a length of 75 points with the first 50 represented by random noise and the last 25 represented by the Ricker wavelet. A random number between 1 and 75 was generated to determine where to sample the training trace. If the random number was less than or equal to 50 then 20 samples were extracted from the trace around the random number. The extracted trace was applied to PEs 1-20. If the random number was greater than 50 then the Ricker wavelet was applied to PEs 1-20. PEs 21-40 always received the Ricker wavelet. The neural network was trained to output a value of 1 when both sets of PEs received the Ricker wavelet and 0 when only one set contained the Ricker wavelet. In test mode, data were extracted from a trace in 20-point increments and the network output a value of 1 whenever a reflected wave appeared in PEs 1-20.

Velocity analysis uses the velocity distribution of p-waves for NMO correction in data processing and interpretation of the seismic data. Since normal moveout is the principle criterion in deciding whether an event observed on a seismic record is a reflection or not, this technique is very important. In data processing the NMO must be eliminated before stacking of the common-midpoint records. One of the most important quantities in seismic interpretation is the change in arrival time caused by dip, and the NMO must be removed before it can be calculated (Sheriff and Geldart, 1995).

In the second stage of processing, Ashida (1996) used a neural network to produce a velocity spectrum using the constant velocity scan method. Synthetic data were generated for a CDP ensemble of 48 folds by a ray tracing method for a horizontal two-layer model. The geophones were at intervals of 25 meters. A Ricker wavelet with a peak frequency of 30 Hz was used for the reflected waves and the internal velocities of the layers were 2000 m/s and 3000 m/s respectively. The rms velocities of the reflected waves from the bottom of the first and second layers were 2000 m/s and 2450 m/s, which were determined by velocity analysis.

The same neural network algorithm used for reflected wave recognition was used for this velocity analysis. The offset axis of the CMP stack was mapped to a velocity axis by stacking the traces in the CMP gather using a constant velocity NMO (Yilmaz, 1987). Since the waveform that was corrected by the most suitable velocity should resemble the original waveform in the CMP gather, a neural network was used to determine which velocity produced the closest match. In other words, a template matching approach using a neural network was used to determine which velocity versus time trace best matched the reflected wave extracted from the CMP stack.

Calderón-Macías et al. (1998) applied a multilayer, feedforward neural network for normal move-out (NMO) correction and velocity estimation that used a more unsupervised approach. Most approaches train the network to recognize relationships between the input seismic signal and known outputs, in this case seismic velocities. However, the seismic velocities are seldom known accurately. Instead, the optimal NMO correction was used to estimate velocities and update neuron weights rather than the error between known and predicted velocities.

Common midpoint (CMP) gathers were transformed to the intercept time and ray parameter domain (τ-p domain) using the cylindrical slant stack method, and they were then used as input to the network. The network found the velocity-time function that provided the best NMO correction. This correction should align all the traces of the CMP gather in phase with the target trace for reflection events. The output of the network was the interval velocities of subsurface layers. These velocities were true interval velocities only in the case of horizontal homogeneous layers.

The network could be trained using data from control locations along a 2-D seismic line. The number of input neurons was determined by multiplying the number of p-traces and the number of samples per trace. The net mapped the input data into output interval velocities or spline coefficients (within velocity search limits) using weighted connections and sigmoid activation functions. The number of velocity layers used to define the velocity-time functions determined the number of output neurons.

The error measure of the NMO correction for a group of CMP gathers was minimized to obtain the optimal group of velocities. Network weights were updated during training using the optimization method of very fast simulated annealing (VFSA). Weight updates were based on estimates from previous iterations and a control parameter called temperature. As cooling occurred in the optimization, only weights which produced a lower error than the previous error were accepted.

Training was complete when the network had output velocities that obtained the best alignment of reflection events in each CMP gather at the control locations. The network was then used for NMO correction and velocity interpolation of CMP gathers at locations other than the control points.

The authors tested this method with both synthetic and real seismic data. A velocity model with anticlinal structure and six velocities ranging from 1.5 to 4.2 km/s was used for the model study. The synthetic data did not include multiples. The network was trained with 11 regularly spaced CMP gathers as input. The hidden layer had 15 neurons and 8 outputs were produced per training example. The network was trained for 2000 iterations, and then applied to intermediate CMPs between control points. The results were accurate except at the extreme edge of the model. The authors then added two control points at each edge of the model and retrained. The CMP gathers were nearly perfectly corrected for NMO and estimated velocities were comparable to true velocities. The authors performed other simulations with varying numbers of CMPs and spacing, and concluded that using several neural networks for different parts of the seismic line was more practical than having one network for the whole line.

Calderón-Macías et al. (1998) next tested the network with surface marine data. The data were pre-processed for spiking deconvolution and multiple attenuation before being transformed to the τ-p domain. Three neural networks were used to process different sections of the seismic line. Each neural network had one hidden layer with 15 neurons and was trained for 3000 iterations. The first network used 20 control CMP gathers for training while the other two used 18. The networks were then applied to 280 CMP gathers along the seismic line, and achieved satisfactory results for both control and intermediate gathers. When the networks were trained with control gathers spaced 0.2 km apart, nearly perfect NMO corrections were obtained. The authors concluded that their method, in which neuron weights are updated on the basis of the quality of the NMO correction instead of the error between known and predicted velocities, is viable and produces accurate results.

6. ELIMINATION OF MULTIPLES

Essenreiter et al. (1997) investigated the removal of multiple reflections in marine seismograms using neural networks. A seismic trace was input into the network which performed deconvolution to recognize and eliminate multiples. The network output the trace with only primary reflection events present. The back-propagation neural network used in this research utilized the RProp algorithm (see Chapter 15 for a description of Rprop).

The network was first trained and tested using a synthetic model that consisted of three deep reflectors below a sea bottom. A set of 200 different seismograms was generated and convolved with a Gabor wavelet. Varying amounts of random noise were added and then the network was trained with 150 of the patterns; the remaining 50 were used for testing.

The network output the edited seismic trace by setting each output neuron to (1) if a primary reflection was present; otherwise output neurons were set to (0). The actual output value was considered the probability of a primary reflection occurring at a certain time. For the synthetic model test data, the network recognized the primary reflections in the following percentage of patterns: 100% for the sea floor, 98% for the first deep reflector, 88% for the second deep reflector, and 78% for the third deep reflector.

The authors then applied this method to real marine, common depth point (CDP) gathers. However, they only had information from one borehole, so the CDP at the well location had to be used for verification (testing). There was no data from other boreholes in the area to use for training; so synthetic well logs were created. This was done by slightly changing the velocity log from the borehole and then inputting it, along with the unmodified density log, into a finite-difference modeling program. Five different synthetic CDP gathers containing multiples were created at five different synthetic well locations. Four were used as input for training the network. The network also required the desired or known output corresponding to these input. For this purpose, synthetic CDP gathers without multiples were generated using Kirchhoff modeling.

The CDP gathers were input to the network one trace at a time. The network consisted of 250 input neurons, one hidden layer with 60 neurons, and 250 output neurons. A total of 300 patterns were used to train the network, which was then tested with the CDP from the well location. The network successfully identified the two deep primary reflections in the seismic data.

7. DECONVOLUTION

Deconvolution can be defined as the process of extracting the reflectivity function from the seismic trace to improve the vertical resolution and recognition of events (Sheriff and Geldart, 1995). Deconvolution operations can be used in a series in which one operation removes one type of distortion and is then followed by a different type of deconvolution operation to remove another. In seismic data processing deconvolution is commonly used to attenuate multiple reflections.

The convolution problem is: Given a source wavelet V_k, and a reflectivity sequence defined by the location of a spike, q, and the amplitude of the spike, r, convolve them to see what the observed seismic response of the earth would be. The deconvolution problem is: Given the observed earth response to an impinging seismic wavelet of unknown form, find the reflectivity sequence that explains the earth response. The deconvolution problem has two parts: 1) define a form for the source wavelet and 2) extract the locations and amplitudes of the reflection events. Inverse filters, Weiner filters, and prediction filters can be used to deconvolve the seismic data and produce the reflectivity sequence. Multiple reflections can be removed from the reflectivity sequence and the edited sequence convolved with the source wavelet to produce an enhanced seismic record.

Wang and Mendel (1992) developed two Hopfield networks for reflectivity magnitude estimation and source wavelet extraction. The two networks were combined in an optimization routine generally known as a block-component method (BCM) for simultaneous deconvolution and wavelet extraction. Since the Hopfield networks minimized the prediction error for a deconvolution process the proposed technique was referred to as adaptive minimum prediction-error deconvolution or AMPED for short. The basic approach was that a source wavelet was computed; an amplitude for a reflectivity series was assumed; spike positions with the assumed amplitude were located; amplitudes at the located spikes were computed; the source wavelet was convolved with the computed amplitudes and subtracted from the trace. The procedure was repeated until amplitudes approaching the noise floor were removed. Wang and Mendel (1992) reported that this technique made no assumptions about the phase of the source wavelet (i.e. the source wavelet does not have to be minimum phase), the type of measurement noise, or whether the reflectivity sequence was random or deterministic.

Calderon-Macias et al. (1997) extended the work of Wang and Mendel (1992) by applying mean field annealing to a Hopfield network to speed convergence. Mean field annealing is similar in concept to simulated annealing but uses deterministic update rules instead of stochastic rules to adjust the variables over time. One of the advantages of the mean field annealing approach is that the normally discrete outputs (0 or 1) of the Hopfield network are replaced with continuous values between 0 and 1. This approach also helps ensure that the Hopfield network converges to the global minimum and not the closest local minimum.

The energy function usually minimized by the Hopfield network was given as equation (5.23). The energy function minimized by the Hopfield network in this example was a modified version given as the error between an observed trace d_k and the computed trace

$$E = \frac{1}{2}\sum_{k=1}^{N}\left[d_k - \sum_{i=1}^{N} V_{k-i}\mu_i + n_k\right]^2, \tag{7.3}$$

where V_k is the seismic source wavelet, with $V_k = 0$ for $k<0$; μ_i is the earth reflectivity sequence; n_k is the measurement noise; N is the number of samples and d_k is the observed trace.

The first step in the processing was to modify equation (7.3) based on a model for the reflectivity sequence defined as

$$E = -\frac{1}{2}\sum_{k=1}^{N}\left[z_k + \sum_{i=1}^{N} V_k\mu_i\right]^2, \tag{7.4}$$

$$z_k = \sum_{i=1}^{N} V_k - i\mu_i + \eta_k, \tag{7.5}$$

and $\mu_i = q_i r_i$, (7.6)

where q takes on binary values of 0 or 1 indicating the presence or absence of a reflection and r represents the magnitudes of the reflections. A separate Hopfield network was used to estimate q and r independently. To estimate q_i we set r equal to a constant, α, and obtained the locations of the reflection events. Then to estimate r_i we used the estimated values of q_i. The values for μ_i were then inserted into equation 7.2 and a Hopfield network was used to minimize E. The value for α was updated and the process was repeated.

Calderon-Macias et al. (1997) and Wang and Mendel (1992) provided the exact energy functions that were minimized and the corresponding weight calculations and input calculations. Both the source wavelet and reflectivity sequence were represented by binary PEs by

$$V_k, r_i = (\sum_{j=1}^{M} \frac{1}{2^{j-1}} p_{ij}) - 1, \tag{7.7}$$

where M is the number of bits of discretization and p_{ij} is either 0 or 1. Equation (7.7) was substituted into the energy equation and used to construct the weight and input vectors.

Normally, the weight update in a Hopfield network is given by

$$x_i(t+1) = f(\sum_{i \neq j=1}^{n} w_{ij} x_j(t)), \tag{7.8}$$

where $x_i(t)$ is the network input at time t at PE i and f is the activation function (normally a step function). Calderon-Macias et al. (1997) reported increased accuracy for the estimation of the reflectivity sequence when they used mean-field annealing to modify equation (7.8) as

$$v_i(t+1) = \tanh\left[\frac{\sum_{i \neq j=1}^{n} w_{ij} v_j(t)}{T}\right] \tag{7.9}$$

where v is a continuous-valued version of the discrete variable x. T is the temperature variable that is gradually reduced during training.

Wang and Mendel (1992) developed a reflectivity estimator given a known source wavelet. Once the reflectivity sequence was estimated it was used, in turn, to estimate the source wavelet by using another Hopfield network. The BCM was developed to allow the reflectivity sequence and source wavelet to be optimized so the final result is the best source wavelet and reflectivity sequence.

8. INVERSION

Recovering a background velocity model from recorded seismograms can be a difficult task. Inversion of the data requires a reliable estimation of the depth-velocity profile. The background velocity is usually obtained from first-break picks or velocity analysis. Roth and Tarantola (1994) investigated an alternative approach whereby a neural network receives a common shot gather as input and outputs the velocity of eight layers of constant thickness plus the underlying half-space. The study did not attempt to analyze field data.

The velocity of the first layer was generated pseudo-randomly as 1,500 m/s +/- 150m/s with a boxcar distribution. Subsequent layers, l+1, had velocities, v, based on the previous layer as generated by

$$v_{l+1} = (v_l + 190) \pm 380. \tag{7.10}$$

The velocities generally increased with depth but some low-velocity zones were allowed. All layers were 200 m thick. A total of 450 models were generated for training purposes and an additional 150 models for testing. Two additional training sets were created with 5% and 10% white noise added. Two additional testing sets were created with 10% and 30% noise.

A MLP network was trained with the back-propagation learning algorithm. The network had 5,420 input PEs since each seismic section consisted of 20 traces sampled at 271 points. The output layer had 9 PEs. The hidden layer contained 25 PEs. The network was trained for 225 epochs.

The test results indicate that the network could predict the velocities of each layer even in the presence of noise although it usually failed to detect low-velocity layers. Even when the predicted velocities were in error, the trend with depth was closely matched.

The work by Roth and Tarantola (1994) is particularly interesting because of the size of the network. No attempt was made to subsample the data. A Cray Y-MP computer was used to train the network but some of today's more powerful desktop computers could handle such a large network. We normally proscribe that the size of the input pattern and hence the number of connection weights needs to be closely coupled with the size of the training set. A heuristic of ten times the number of samples as connection weights dictates that input patterns of the size used by Roth and Tarantola would require millions of training models. Extensive testing by Roth (1993) indicated no improvement in results as the training set size exceeded 200 models. Training set design remains a trial and error process and is highly data dependent. It remains difficult or impossible to predict a priori how many training samples are required to give the best results but the work presented in Chapter 17 sheds some light on ways to limit the number of samples required based on model sensitivity analysis.

Another method of performing seismic inversion is described in Chapter 12 and includes a case study.

REFERENCES

An, G., and Epping, W., 1993, Seismic first-arrival picking using neural networks: World Congress on Neural Nets, INNS, **I**,174-177.

Ashida, Y., 1996, Data processing seismic data by use of neural network: Journal of Applied Geophysics, **35**, 89-98.

Calderon-Macias, C., Sen, M., and Stoffa P., 1997, Hopfield neural networks, and mean field annealing for seismic deconvolution and multiple attenuation: Geophysics, **62**, 992-1002.

Chen, Q., and Sidney, S., 1997, Seismic attribute technology for reservoir forecasting and monitoring: The Leading Edge of Exploration, **16**, 445-456.

Chu, C., and Mendel, J., 1994, First break refraction event picking using fuzzy logic systems: IEEE Transactions on Fuzzy Systems, **2**, 255-266.

Dimitropoulos, C., and Boyce, J., 1993a, Applications of neural nets to seismic signal analysis: Int. Con. on Acoustic Sensing and Imaging, IEE **369**, 62-67.

Dimitropoulos, C., and Boyce, J., 1993b, Neural nets for first break detection in seismic reflection data: Third International Conference on Artificial Neural Networks, IEE **372**, 153-172.

Dimitropoulos, C., and Boyce, J., 1994, First break detection in seismic reflection data with fuzzy ARTMAP neural networks: Signal Processing VII, Theories and Applications, Proceedings of EUSIPCO-94, Seventh European Signal Processing Conference, Lausanne, Switzerland: Eur. Assoc. Signal Process, **1**, 221-224.

Essenreiter, R., Karrenbach, M. and Treitel, S., 1997, Elimination of multiple reflections in marine seismograms using neural networks: 1997 IEEE International Conference on Neural Networks, New York, NY:IEEE, **4**, 2157-2161.

McCormack, M., Zaucha, D., and Dushek, D., 1993, First-break refraction event picking and seismic data trace editing using neural networks: Geophysics, **58**, 67-78.

Murat, M., and Rudman, A., 1992, Automated first arrival picking: a neural network approach: Geophysical Prospecting, **40**, 587-604.

Palaz, I., and Weger, R., 1990, Waveform recognition using neural networks: Geophysics: The Leading Edge of Exploration, **9**, 28-31.

Roth, G., and Tarantola, A., 1994, Neural networks and inversion of seismic data: Journal of Geophysical Research, **99**, 6753-6768.

Roth, G., 1993, Application of neural networks to seismic inverse problems: Ph.D. Dissertation, University of Paris 7, Paris, France.

Sheriff, R., and Geldart, L., 1995, Exploration Seismology: Cambridge University Press.

Veezhinathan, J., and Wagner, D., 1990, A neural network approach to first break picking: IJCNN International Joint Conference on Neural Networks, New York, NY: IEEE, **1**, 235-240.

Veezhinathan, J., Wagner, D., and Ehlers, J., 1991, First break picking using a neural network, *in* Aminzadeh, F. and Simaan, M., Eds., Expert Systems in Exploration: Society of Exploration Geophysicists, 179-202.

Wang, L., and Mendel, J., 1992, Adaptive minimum prediction-error deconvolution and source wavelet estimation using Hopfield neural networks: Geophysics, **57**, 670-679.

Yilmaz, O., 1987, Seismic Data Processing: Society of Exploration Geophysicists.

Chapter 8
Rock Mass and Reservoir Characterization

Mary M. Poulton and Kathy S. Powell

1. INTRODUCTION

Neural networks can play an important role as an interpretation aid for rock mass and reservoir characterization prior to and during production. The use of self-organizing neural networks for facies mapping has been commercialized and successfully applied in several fields. We will increasingly see neural networks play a role in time-lapse seismic and multi-component interpretation. Neural networks also offer the potential of real-time interpretation of drilling sensors and borehole logs for rock mass characterization. Neural network technology will undoubtedly play a role in the "e-field" concept where nearly all aspects of production are monitored and controlled in real-time. In this chapter we will review neural network applications to facies mapping, time-lapse seismic interpretation, log property prediction, and reservoir characterization.

2. HORIZON TRACKING AND FACIES MAPS

Seismic data must ultimately be related to the underlying geology, especially stratigraphic layers or structures with economic potential. Rocks with similar physical properties at approximately the same depth should yield similar seismic signatures. Mapping these horizons can be accomplished by grouping similar seismic signatures into the same class. If the classification is unsupervised, similar to K-means clustering, similar signatures are placed in the same class and the interpreter must assign some meaning or label to the class (e.g. porosity values). If the classification is supervised, the interpreter defines the desired characteristics of a seismic signature and the computer software labels all similar examples in the data set. Horizon picking is part of the structural analysis of a three dimensional dataset. The interpreter must combine knowledge of the overall regional geology and of stratigraphic and structural relationships with the character of the seismic reflection to determine the events that can be grouped as the same horizon. Most commercial autotracking software programs use a dense grid of points that are input by hand to be able to track the horizons accurately. These traditional methods use classical pattern recognition methods, such as wavelet recognition and cross-correlation, and rely on a-priori knowledge of seismic arrivals which are fairly limited. Leggett et al. (1996) described a horizon tracking method using a combination of supervised and unsupervised learning with a hybrid SOM and MLP network.

The term facies refers to the sum total of features that characterize the environment in which sediment is deposited (Sheriff and Geldart, 1995). Facies include sedimentary structure, bedding form, attitude, shape, thickness, and continuity variations in sedimentary units. Seismic facies maps are the results of organizing and recognizing seismic reflection patterns in reflection data. This type of analysis is called seismic facies analysis and is used to extract nonstructural information from seismic data. The patterns are classified and grouped by slow visual identification, plotted, and then used in exploration. The maps can have large variances between interpreters due to the manual nature of the task. Addy (1998) describes an unsupervised SOM network for facies mapping.

Leggett et al. (1996) developed an automatic horizon tracker, which enabled horizons to be tracked in three dimensions with little input from the interpreter. They used a hybrid neural network that had a combination of unsupervised and supervised learning capabilities (Self Organizing Map and the MLP). The hybrid network was trained with examples of horizons that originate from one 3D data volume. After the training, the tracker was able to recognize and classify the horizons across lines and traces of other datasets.

Training data were extracted from a window of seismic data centered on the horizon of interest. The SOM was trained to activate the same or spatially near PEs when signatures with similar morphology were presented. Hence the SOM acted as a filter to extract the common features of several data samples (see Chapter 10 for an example). Once the SOM had been trained, its PEs were used as input to a MLP. The purpose of the MLP was to assign a class label to similar patterns. While the SOM was trained solely on data from the horizon, additional training data not from the horizon were added to the training set when the MLP was trained. The SOM activated different PEs for the non-horizon data and the MLP received a different pattern of active PEs from the SOM. While the MLP might have been able to perform the classification alone, the addition of the SOM as a pre-processing step generally simplified the problem.

The network was tested using a 3D seismic dataset with line and crossline spacing of 25 meters from the North Sea. Two horizons were mapped in a structurally complex area. Each horizon required its own training set. The first horizon had a training set of 90 samples. The SOM had 200 PEs and the MLP had 46 input PEs, 96 PEs in the first hidden layer, 48 in the second hidden layer and 2 in the output layer. The second horizon used 80 training samples. The SOM was the same size but the MLP had 62 input PEs, 24 hidden PEs in each layer, and 2 output PEs.

Addy (1998) proposed that the information contained in seismic traces could be efficiently extracted and correlated with geologic information by using an unsupervised neural network classification on traces in specific time windows. Similar looking traces should represent similar geology. Therefore a thematic map produced from classification of trace shapes can be correlated with a seismic facies map used to yield geological information.

A SOM network performs an unsupervised classification in the Stratimagic® software by Flagship Geo (a subsidiary of Compagnie Générale de Géophysique). Addy (1998) presented a case history from South Texas where the application of neural networks accentuated subtle geological features through the classification of the seismic traces. Addy (1998) reported that the technique allowed them to "identify the porosity in the Edwards limestone, define the Sligo reef trend, map complex channel systems in the upper Wilcox and Frio sections and determine the extent of the lower Wilcox onlapping sands in the Lavaca channel." A similar method is presented in Chapter 10 and the training procedure is described in more detail.

Poupon et al. (1999) used neural networks to analyze variations of seismic trace shape associated with a seismic facies map for reservoir characterization. Their methodology was based on seismic facies analysis in conjunction with litho-seismic modeling from well curves. Seismic facies maps of the regional depositional environment were used to analyze gas sands associated with a channel system. The seismic trace shapes associated with a gas-producing well and a non-producing well were then integrated into the litho-seismic model for calibration (training). Seismic responses associated with variations of sand thickness, reservoir porosity and fluid content were generated and compared to traces generated by the neural network. The authors concluded that this methodology accurately and quickly characterized the gas sand reservoir in the study area.

3. TIME-LAPSE INTERPRETATION

In-fill drilling to find by-passed oil can be guided by seismic surveys acquired before and during production. The 4D seismic surveys look for changes in the interaction between the reservoir rock and pore fluids during drainage. Oldenziel et al. (2000) used an MLP with back-propagation learning to predict porosity and water saturation in the Statfjord field. Production in the field started in 1979. The study used 3D surveys from 1991 and 1997 to predict the changes in porosity and water saturation.

Synthetic seismograms were constructed for five volumes: mid- and far-angle reflectivity; mid- and far-angle elastic impedance; and acoustic impedance (Oldenziel et al., 2000). The target values for water saturation and porosity were constructed from logs. The network was given the full waveform in a 150 ms window from each processed volume as input. The 10 input volumes were combined and then mapped by separate MLPs to porosity and water saturation volumes for intervals in the Brent Group. In addition to the seismic data, the reference time relative to the top of the Brent interval and a (0,1) flag for the East and West flank of the interval were added to the input pattern. The network that predicted water saturation included the predicted porosity from the previous network as an input.

The networks were tested on the real seismic data from the 1991 and 1997 surveys. The predicted water saturation curves for the synthetic training data had lower frequency content than the log data, which was expected for synthetic data (Oldenziel et al., 2000). The saturation curves for the field data were a closer match to the log data. Both the horizon and stratigraphic slices were able to accurately map the reservoir drainage patterns and areas of depletion.

4. PREDICTING LOG PROPERTIES

Gathering borehole-derived data is one of the most expensive aspects of any exploration program. A judicious combination of surface geophysical data and limited well data can increase our knowledge of the subsurface if we reliably transform from one domain to the other. Specifically, we would like to be able to predict more expensive well log measurements from less expensive surface-collected data. If we have sonic log data, we can use Wyllie's equation to estimate porosity. Hence, if we can use the seismic data to predict sonic log data at any point in the seismic volume, we can produce porosity maps for the entire volume using only a small number of wells for control.

Liu and Liu (1998) extracted seismic traces from the vicinity of wells and used the traces to predict the well parameters of sonic velocity, density, and shale content. The training set for their MLP required that the input data compensate for the fact that well log data had higher frequency content than the seismic data. The input data therefore consisted of seismic amplitudes within a time window and the low-pass-filtered sonic log values. The output data to be predicted were the sonic velocities, densities, and shale content for each sample point in the window.

In the first example that Liu and Liu (1998) presented, they used three wells from an oil field in China. The sonic and density logs were used to calculate the reflection coefficient series. The reflection series was convolved with a 50 Hz wavelet to produce a synthetic seismogram. The synthetic seismic data were used as input to the neural network along with the 0-10 Hz component of the sonic log data. The network was trained until it could predict the sonic velocities, densities and shale content with an overall accuracy of 95%. The network was then tested using data from the other two wells. The density estimates were the worst fit of the three output parameters largely because of the low-frequency nature of the density log.

Following the test using synthetic seismic data, real migrated data were used as input. Wells 2 and 3 from the previous example were used for training and well 3 was used for testing. Five seismic traces adjacent to each well were extracted and used as input in addition to the low-pass filtered sonic log data. The xy location of each well was used as an input and the times for each sample of the seismic traces were also included as part of the input pattern vector. Only the sonic velocity and shale content were estimated. When the sonic velocities and shale contents were estimated across the extent of the seismic profile, lithologic boundaries became very clear.

Collaboration between Hampson-Russell Software, Ltd., and Mobil E&P Technical Center has led to the commercial development of a reliable way to predict log properties from seismic attributes (Russell et al., 1997; Hampson et al., 1999). The basic premise is that while it would be desirable to only use attributes that have a known physical relationship with a rock property, we often need to consider attributes that have a statistical relationship as well. The statistical relationship can be expressed, for example, through a linear weighted sum, such as

$$\Phi(x,y) = w_o + w_1 I(x,y) + w_2 E(x,y) + w_3 F(x,y), \tag{8.1}$$

where the porosity, Φ, is a function of the acoustic impedance I, amplitude envelope E, and instantaneous frequency F (Russell et al., 1997). A linear multiple regression approach could be used to map the attributes to the porosity or a robust non-linear technique, such as computational neural networks.

The software package EMERGE® by Hampson-Russell Software, Ltd. uses a variation on the probabilistic neural network, called the generalized regression neural network, to map seismic attributes to well parameters. The user interactively selects a window of data to analyze. Seismic traces in the vicinity of one or more wells are extracted, averaged and then user-selected attributes are applied to the composite trace. The attribute values are used as the input pattern and the output is the desired well parameter, such as porosity. A boot-strap or cross-validation technique is used for validation meaning that one well is removed from the training set and used for testing; then, it is placed back in the training set and a different well is removed.

The effectiveness of this approach depends on how well the selection of attributes can predict the log values and how accurately the neural network can be trained. Hence, Hampson et al. (1999) analyze the predictive power of combinations of 17 different attributes (see Chapter 10 for a list of attributes). The results presented for the University of Calgary CREWES Project dataset from the Blackfoot area of Western Canada and the Pegasus Field in West Texas indicate that the probabilistic neural network provides sufficiently accurate estimates of well parameter values and very good resolution of thin beds.

Todorov et al. (1998) used multiple seismic attributes from a 3C-3D seismic survey to predict well logs using a non-linear, statistical relationship. Step-wise regression was used to determine the optimal set of seismic attributes to input to a neural network for sonic velocity prediction. For the example cited the attributes included impedance, integrated trace, time, instantaneous phase, amplitude envelope, and amplitude weighted frequency. Applying a convolutional operator to each of the attributes modified conventional regression. In cross-validation tests, the neural network results showed a high prediction correlation of 0.88 compared to 0.85 for multi-regression analysis.

Hampson and Todorov (1999) implemented a similar methodology for AVO lithology prediction. However, a suitable amount of well data for training the neural network was not available. Synthetic training data were created using Biot-Gassman modeling to make pseudo-wells from a single well using fluid substitution. AVO attributes were calculated for the synthetic CDP gather at each well. A probabilistic neural network was able to predict Poisson's Ratio using seven pre-stack AVO attributes.

Walls et al. (1999) identified reservoir lithologies using neural networks and core, well log, and post-stack seismic data. The neural network was first trained with well log data from five wells to predict lithology in a reservoir. The network was input lithology from one well, along with density, total porosity, V_p, V_s, clay volume, and water saturation.

Synthetic seismograms were generated at each of the well locations to correlate the well log attributes and seismic attributes. The reservoir classification curves generated in the first step were converted to time using the synthetic seismogram derived time-depth curve.

Seismic attributes were calculated from the synthetic seismograms and used with the time-domain lithology classes to train the neural network to predict lithology classes throughout the seismic data volume. Walls et al. (1999) initially calculated 16 seismic attributes for analysis, but the use of five attributes gave optimal results. Of the five wells in the survey area, the two producing wells were classified as being located in the oil sand area and the three non-producing wells were classified as being outside of this area.

5. ROCK/RESERVOIR CHARACTERIZATION

Link (1995) conducted a preliminary study in which back-propagation neural networks were used to predict lithology and reservoir characteristics. The first part of the study involved using well log suites over a target range of depths to predict paleosol sequences in a well. Then resistivity, gamma, and density logs were used to predict sonic and porosity well logs. Finally, 3-D seismic data were input to a BPNN that attempted to predict the spatial distribution of porosity in a reservoir.

The study site for the reservoir characterization was approximately one square mile in area and contained twenty-one wells, eleven of which were producing oil. The neural network was trained and tested using 3-D seismic data (with dynamite as the source) as input. Output consisted of porosity log values.

A seismic trace from a common midpoint location closest to a particular well was selected. A time window corresponding to the producing interval of the reservoir was used to extract values from the seismic trace to use as input. The porosity values in the corresponding depth interval from that well's porosity log were used as the desired output.

The neural networks used by Link (1995) had one hidden layer containing ten PEs. The number of input PEs was determined by the number of samples in the time window of seismic data. The number of output PEs was determined by the number of values in the porosity log interval.

The network was trained and then tested using data from the 3-D seismic data cube. Although the results of this first network were not given, Link (1995) attributed the poor performance of the network to the prohibitively large number of connections between PEs. Link (1995) then tried several variations of the input seismic data and output porosity data. The best results were obtained using seismic data that had been high-pass filtered prior to being input to the network. Including instantaneous phase as input, along with the raw seismic data, improved some of the porosity predictions but not all. Using average porosity values instead of the full porosity log for training did not improve results. Link (1995) concluded that the back-propagation network could predict the general trend of the porosity well logs, but results for individual values of porosity had a large prediction error.

Yoshioka et al. (1996) evaluated the performance of kriging, neural networks, and cokriging to predict the porosity thickness of a reservoir. Simple kriging was performed using only well data. The neural network analysis used both well data and 21 seismic attributes from 3-D seismic data. The cokriging method used the output of the neural network.

The study used log data from 42 wells, 27 of which were deviated. The following seismic attributes were extracted from a 3-D seismic data set using a time window of 20 ms between the upper and lower horizons of the reservoir:

Amplitude Statistics
RMS amplitude
Average absolute amplitude
Maximum peak amplitude
Average peak amplitude
Maximum trough amplitude
Average trough amplitude

Complex Trace Statistics
Average reflection strength
Average instantaneous frequency
Slope of reflection strength
Slope of instantaneous frequency

Sequence Statistics
Energy half-time
Ratio of positive/negative

The following spectral statistics were extracted from a time window of 40 ms above the lower horizon to ensure a sufficient number of samples:

Spectral Statistics
Peak spectral frequency
Spectral slope from peak to maximum frequency
Dominate frequency series F1
Dominate frequency series F2
Dominate frequency series F3

In addition to these attributes, the travel time and amplitude of the lower horizon and X-Y coordinates were used as input to the neural network.

The neural network consisted of 21 input PEs, a hidden layer with 14 PEs, and one output PE to predict the porosity thickness. The training data set contained the seismic attributes from the 42 well locations as input, and the well log derived porosity thickness as output. The error between the network-estimated value and the well log derived value for the porosity thickness was less than 1% for the training data set. After the network was trained, it was applied to the entire gridded area of the seismic survey.

The neural network estimated values for the survey grid were then used as soft data for cokriging. Yoshioka et al. (1996) modeled the spatial connectivity function in the range h ⟨100 m for lag. The cokriged distribution for the porosity thickness was more heterogeneous and had a smaller standard deviation than the kriged values.

The authors performed two cross-validation tests for each of the three methods of predicting the porosity thickness of the reservoir. For the first test, data from one of the 42 wells, was not included in the analysis/training. The estimated value at the hidden well location was then checked for each of the three methods. Each of the 42 wells was hidden in turn. The neural network method showed the largest dispersion between its estimated values and the actual ones. The dispersion for kriging and cokriging was similar. The error exceeded the standard deviation in 23 cases for kriging, and in 24 cases for cokriging.

For the second cross-validation test, Yoshioka et al. (1996) used data from 15 vertical wells for analysis/training; the 27 hidden well locations were used to evaluate the estimated porosity thickness for each method. The dispersion between actual and estimated values was again highest for the neural network method, while kriging and cokriging results were similar. The error exceeded the estimated standard deviation at 11 hidden well locations for kriging, and 15 locations for cokriging. The authors concluded that cokriging using neural network output can provide plausible porosity thickness distribution maps, but the accuracy of this method was still less than kriging using only well data. One reason for this was that the resolution of the seismic survey was too coarse for the thin reservoir layer, thus there may have been poor correlation between the seismic and well log data.

REFERENCES

Addy, S., 1998, Seismic facies maps: A quick exploration tool: AAPG Bulletin, **82,** 522.

Hampson, D., Schuelke, J., and Quieren, J., 1999, Use of multi-attribute transforms to predict log properties from seismic data: submitted to Geophysics.

Hampson, D., and Todorov, T., 1999, AVO lithology prediction using multiple seismic attributes: 69th Annual International Meeting, Society of Exploration Geophysicists, Expanded Abstracts.

Leggett, M., Sandham, W., and Durrani, T., 1996, 3D horizon tracking using artificial neural networks: First Break, **11**, 413-418.

Link, C., 1995, Artificial neural networks for lithology prediction and reservoir characterization: Proceedings of the SPIE – The International Society for Optical Engineering, **2571**, 163-174.

Liu, Z., and Liu, J., 1998, Seismic-controlled nonlinear extrapolation of well parameters using neural networks: Geophysics, **63**, 2035-2041.

Oldenziel, T., de Groot, P., and Kvamme, L., 2000, Statfjord study demonstrates use of neural network to predict porosity and water saturation from time-lapse seismic: First Break, **18**, 65-69.

Poupon, M., Azbel, K., and Palmer, G., 1999, A new methodology based on seismic facies analysis and litho-seismic modeling, The Elkhorn Slough Field Pilot Project, Solano County, California: 69th Annual International Meeting, Society of Exploration Geophysicists, Expanded Abstracts.

Russell, B., Hampson, D., Schuelke, J., and Quirein, J., 1997, Use of multi-attribute transforms to predict log properties from seismic data: The Leading Edge of Exploration, **16**, 1439-1443.

Sheriff, R., and Geldart, L., 1995, Exploration Seismology: Cambridge University Press.

Todorov, T., Stewart, R., Hampson, D., and Russell, B., 1998, Well log prediction using attributes from 3C-3D seismic data: 68th Annual International Meeting, Society of Exploration Geophysicists, Expanded Abstracts.

Walls, J., Taner, M., Guidish, T., Taylor, G., Dumas, D., and Derzhi, N., 1999, North Sea reservoir characterization using rock physics, seismic attributes, and neural networks: a case history: 69th Annual International Meeting, Society of Exploration Geophysicists, Expanded Abstracts.

Yoshioka, K., Shimada, N., and Ishii, Y., 1996, Application of neural networks and cokriging for predicting reservoir porosity-thickness (SIGMA oh): GeoArabia, **1**, 457-470.

This Page Intentionally Left Blank

Chapter 9
Identifying Seismic Crew Noise

Vinton B. Buffenmyer

1. INTRODUCTION

Seismic crew noise is a type of seismic interference that occurs when competing marine crews are actively shooting surveys in the same area. The shot noise recorded from the competing surveys can be removed if the crews practice time-sharing of shots and exchange navigational information (Hargreaves et al., 1997; Manin and Bonnot, 1993). If a cooperative agreement cannot be reached, there exists no easy way to remove the shot-noise generated by rival crews.

The noise patterns created by this type of interference are easily recognized by human interpreters in the shot records but the volume of data recorded in a typical 3D survey is too large for manual processing to be time or cost effective. Within an individual seismic trace, the interference consists of isolated, short duration events that are unremarkable. Across several traces, however, a characteristic trend is developed that is dissimilar to the trends of the true data representing real geologic boundaries (Figure 9.1) (Buffenmyer, 1999). It is this characteristic trend that allows the interference to be discriminated by the neural network.

I will focus on the sensitivity of the neural network to training set design in this chapter. Normally good training set design dictates nearly equal representation from all classes in the data set but in this example I want to introduce some bias into the training set by not using an equal representation. It was discovered that for every time the network tested poorly, changes in the training set could improve the results. Therefore, the inclusion of certain patterns and their distribution in the training set had to be carefully monitored.

As was noted in Chapter 3, training statistics alone do not fully predict performance of a neural network. I will show in this chapter that there are cases where the best indicator of performance is a visual display of results rather than a global statistic.

1.1. Current attenuation methods

There are several methods currently being used to avoid, or subsequently attenuate, seismic crew noise interference. In theory, the simplest solution is to use time-sharing when several crews are shooting surveys in a congested area. Each crew acquires data only within a specified time period thus avoiding shot noise from other crews. This method requires diplomacy between rival crews and can significantly increase both the cost and duration of the survey.

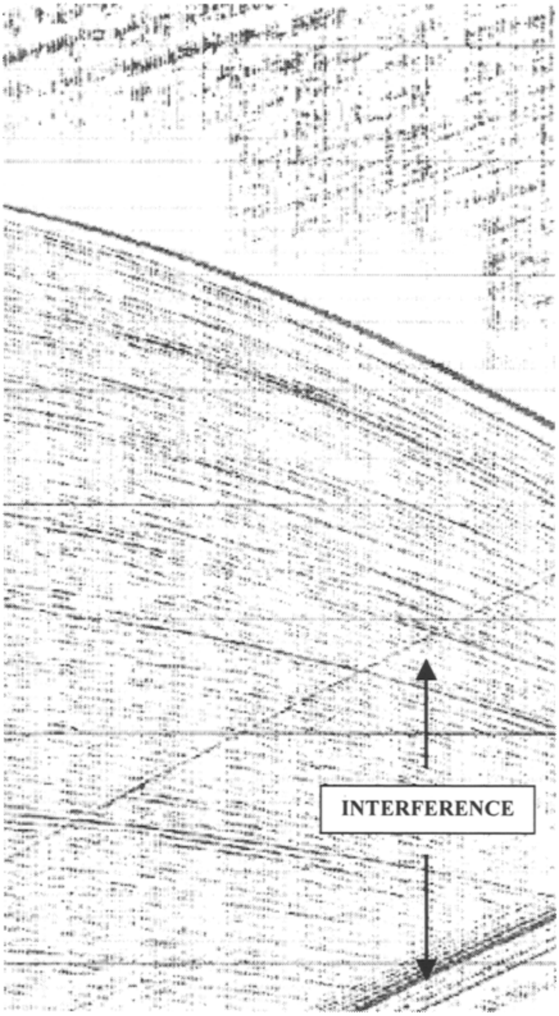


Figure 9.1. Shot display exhibiting seismic crew noise interference patterns arriving with opposite moveout. No NMO correction applied.*

An alternative approach to time-sharing is the prediction and filtering of crew noise following the exchange of location and shooting information between rival crews. Each crew shoots their survey simultaneously and then exchanges information about their location and timing. This cooperation requires that rival crews exchange position information, synchronize their GPS clocks, have different shooting cycles, and avoid shooting broadside to other vessels (Hargreaves et al., 1997).

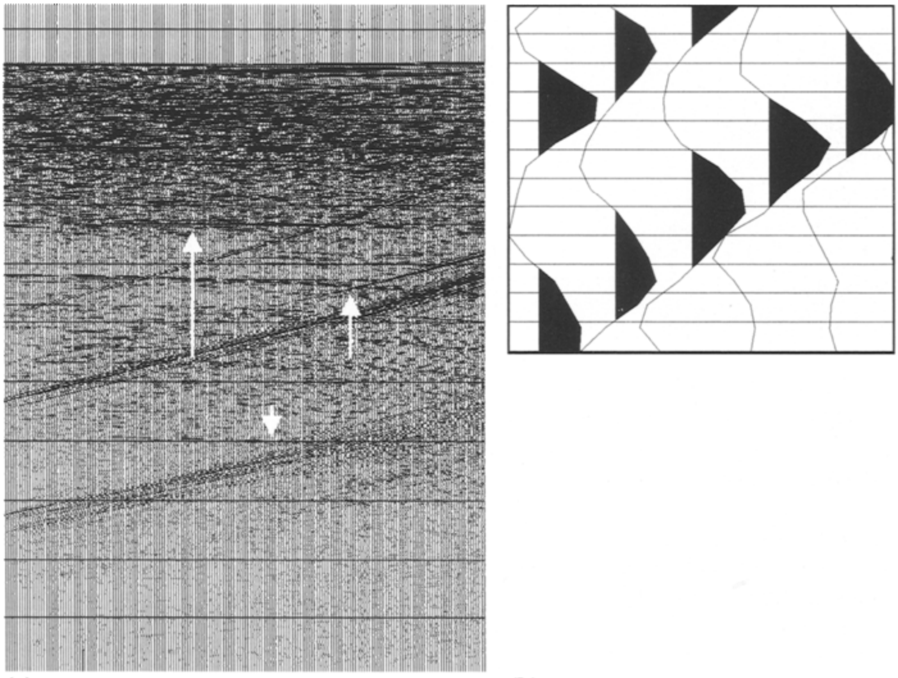
---

* Data were provided by CGG Americas, Inc. and TGS, Inc.

By knowing the relative coordinates and firing times of the interfering vessels, it is possible to predict the arrival of crew noise at the recording streamer. With the predicted arrival times, it is possible to apply a time shift and effectively flatten the noise within the shot display. Once flattened, the noise can be removed by applying a dip filter, such as an *f-k* filter. A reverse time shift is applied and an edited shot display produced. Further attenuation can be achieved by applying FX deconvolution to receiver, constant offset, or CMP gathers (Canales, 1984). If the organization of the crew noise interference has been sufficiently disrupted, then the CMP stack will further remove the random noise (Lynn et al., 1987).

1.2. Patterns of crew noise interference

Seismic crew noise can be seen in shot displays in three distinctly different patterns each dependent upon the relative locations of the recording and interfering vessels. The most easily distinguishable type of pattern is rear-end interference. This occurs when the interfering vessel is located behind the recording vessel and the interference arrives with a moveout opposite from that of the true signal. After normal moveout (NMO) velocities are applied, the true signal generally flattens out depending on local geology, but the rear-end interference appears to dip towards the near offset, or to the left in the examples given. For convention, this will be defined to be a negative dip. Examples of a shot display and a training pattern containing rear-end interference are shown in Figure 9.2. Note that even at the small scale of the five-trace by 60-millisecond window, the negative sloping pattern can still be recognized.



(a) (b)

Figure 9.2. (a) A shot display containing rear-end interference and (b) an extracted window, measuring five traces by 60 milliseconds, used for training.

A slightly more difficult pattern is that of front-end interference. In this scenario, the interfering vessel is in front of the recording vessel and the interference arrives with a similar moveout to that of the true data, varying only slightly in slope. In this case, the benefits of applying an NMO correction become apparent. Prior to the correction, both signal and interference have positive dips, increasing in time towards the far offset. Applying NMO velocities allows the two patterns to separate further, as the signal flattens out and the front-end interference remains positively dipping (Figure 9.3). Figure 9.4 shows an example of a front-end interference-training pattern.

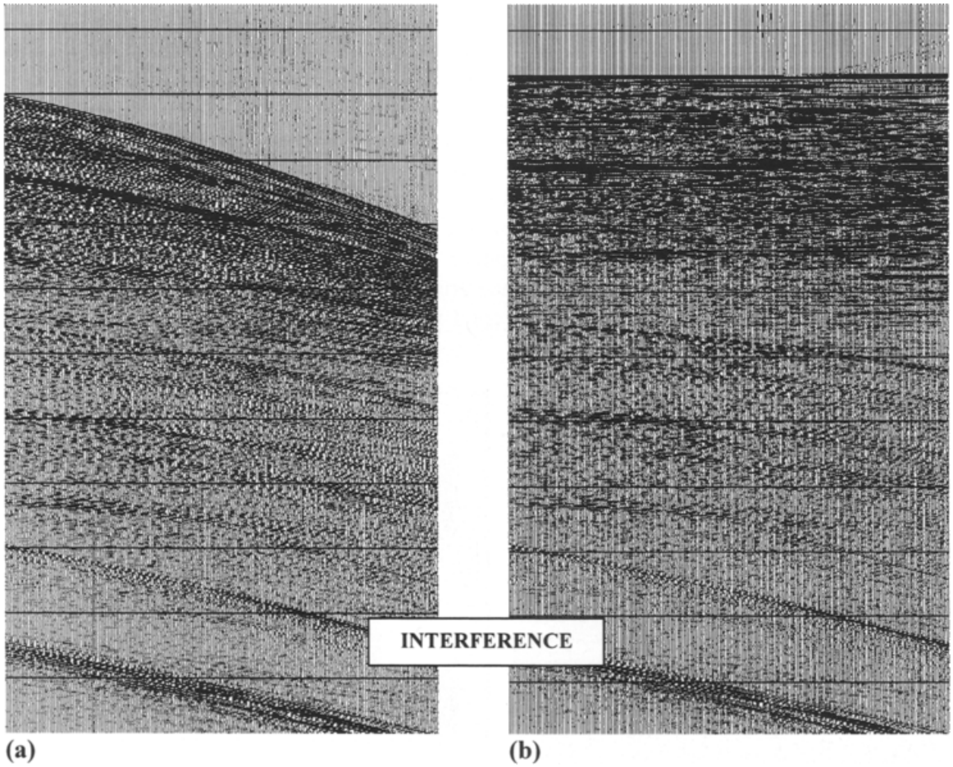


Figure 9.3. Shot displays with front-end interference. (a) is without NMO correction and both signal and interference show dip. Interference has a more dissimilar pattern after NMO correction is applied to (b).

The third, and most difficult, pattern to distinguish is sideswipe interference. This occurs when the two vessels are side by side or when the interfering vessel is moving in a direction normal to the recording vessel. The apparent moveout of the interference is similar to that of the true signal. In this case, NMO corrections do little to differentiate the interference from the signal (Figure 9.5). The curvatures of their hyperbolas may vary, but it often is not enough to safely remove one without removing the other. This becomes an area of limitation, where the neural network cannot accurately identify the interference. This area of limitation is believed to occur when the interfering vessel is between +15° and -15° of the normal to the streamer of the recording vessel (Hargreaves et al., 1997). In my example, the network is not trained on sideswipe interference that appears similar to that of true signal. Instead, it is trained on cases of rear-end and front-end interference where the patterns are slightly dipping and approaching horizontal. It is important to draw a cutoff point where the interference becomes too close to horizontal and cannot be differentiated from signal.

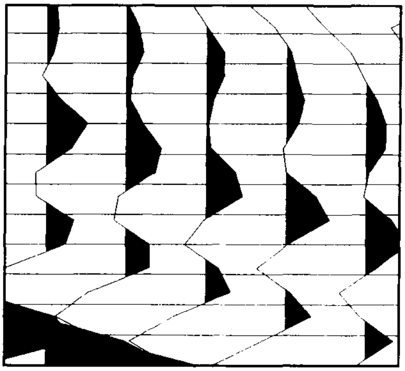
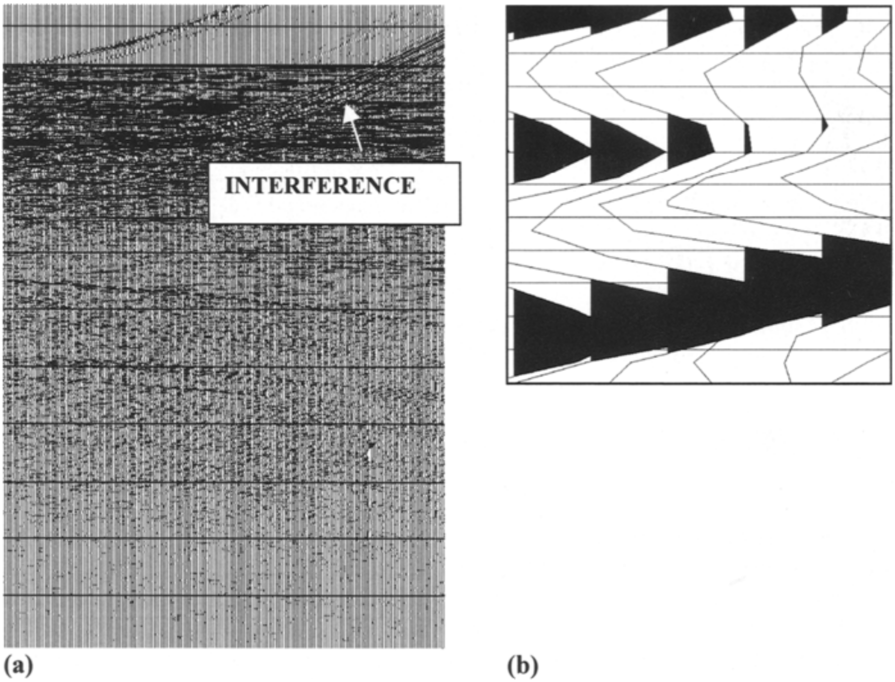


Figure 9.4. Extracted window of front-end interference used for training. Window size is five traces by 60 milliseconds.



(a) (b)

Figure 9.5. (a) Shot display exhibiting sideswipe interference and (b) its extracted window with pattern approaching horizontal. This type of window is not used for training due to its ambiguity.

1.3. Pre-processing

Each shot record that was selected for neural network training and testing was processed identically to create a uniform starting point. Because amplitude versus offset (AVO) analysis was to be performed on the data and required relative amplitude values, only reversible processes were used. The data were corrected for the normal moveout (NMO). A known velocity function for the region was used to make the corrections. To make the crew noise identification as rapid and simple as possible for potential on-ship processing, we intentionally chose to do no other processing after the NMO correction.

Following the NMO correction, shot records with examples of front and rear interference were chosen for training. Since a shot record that is 14,000 milliseconds long and 288 traces wide, with a four-millisecond sampling rate, contains 1,008,288 amplitude values, some method of subsampling the record was necessary. An appropriate window size was determined following analysis of its effect on both the accuracy of pattern recognition and the size of the neural network. There was an inherent trade-off between the two. The window needed to be large enough to encompass a pattern and allow it to be accurately identified by the network. Conversely, it needed to be small enough to keep the network architecture moderate in size and quick in processing speed. An appropriate window size was found to be five traces wide by 60 milliseconds long, a significantly smaller subset of the entire shot record.

Before the smaller windows could be used for training and testing, a standard way of recording their locations within the shot record had to be implemented. A coordinate system was developed to locate each window in the time and offset (t,x) plane. This was accomplished by using the trace number and two-way travel time of the upper left-hand corner of each window. For example, the window identified as 56, 6700 covers traces 56 to 60 from times 6700 to 6760 milliseconds.

Each window had to be classified as either signal or interference, depending upon which type of pattern it contained. Patterns that appeared horizontal were classified as signal and given a desired output of 1. Patterns that were not horizontal, both with a positive or negative slope, were identified as interference and given a desired output of -1. Windows that contained both signal and interference patterns were classified as interference, while ambiguous patterns were simply not included in the training set.

2. TRAINING SET DESIGN AND NETWORK ARCHITECTURE

For the purpose of this study, a MLP architecture with back-propagation learning was chosen because of its relative simplicity. We also tried a radial basis function and modular neural network architecture but did not find any improvement in the results. The best network was found to have 75 input PEs, 13 hidden PEs, and one output PE. The hyperbolic tangent transfer function was used to modify activations in the hidden layer. A connection strategy was chosen that directly connected the input layer to the output layer in addition to the hidden layer. The different connection strategy provided better accuracy than the more conventional strategy of only connecting the hidden layer to the output layer.

The two factors that had the most influence on the overall accuracy of the classification were the training set design and the schedule used to change the learning rate and momentum during training. The hidden layer could have between 10 and 20 PEs and still result in the same overall accuracy. The number of misclassifications increased with fewer than 10 and more than 20 PEs.

The schedule used for adjusting the step size and momentum during training is shown in Table 9.1. The initial values for step size and momentum were specified and then reduced by a factor of 0.95 for each set of iterations. The iteration schedule was created to emphasize adjustments early in training and was determined by trial and error.

Table 9.1
Schedule used to adjust training parameters

	Hidden Layer				
Iterations	1,000	3,000	7,000	15,000	31,000+
Step size	0.3	0.285	0.257	0.209	0.139
Momentum	0.8	0.76	0.686	0.559	0.371
	Output Layer				
Step size	0.15	0.142	0.128	0.105	0.069
Momentum	0.8	0.76	0.686	0.558	0.371

2.1. Selection of interference training examples

A collection of five-trace by 60-millisecond windows of data was used as the training set, which was intended to represent all the different types of crew noise interference that could occur in marine surveys as well as those patterns associated with signal.

For this particular classification problem, it was necessary to first determine what seismic attributes could be used to discriminate between the desired true signal and the unwanted crew noise interference. Once these attributes were decided upon, the feasibility of extracting them and presenting them to the network had to be considered. Attributes requiring large amounts of processing time or memory were dismissed in favor of those that were readily attainable and easy to manipulate. Because there were 1,008,288 samples per shot record, a smaller window had to be chosen for extracting and presenting data to the network. The chosen attributes had to be capable of showing a pattern within the smaller window.

Using these attributes for training, one must determine how many different ways they can be arranged to represent either signal or noise. Does the noise always appear in the same manner, or does it change from one shot record or survey to the next? It is important that, for every possible arrangement of attributes, the network be amply trained on similar arrangements. This will provide the network with the experience necessary to make correct decisions about new data in the future.

In keeping with the belief that the simplest answer is often the best answer, it was decided that raw amplitude values could be used to best represent the seismic crew noise interference to the neural network. No gain was applied, and no transformations were performed. All data were taken directly from the SEG-Y formatted tape and used in the time and offset (t,x) domain as seen in a raw shot record. Only NMO corrections were applied using velocities from an accurate velocity library. Figure 9.6 shows a portion of a shot display used for training.

The amplitude values were extracted from the shot record using a window that was five traces wide and 60 milliseconds long. Even in a window this small, the horizontal pattern of signal was distinguishable from the dipping pattern of the interference (Figure 9.7). Though not explicitly presented to the network, the frequency of the events captured within the window became a factor. Extremely low-frequency events have a pattern that may be difficult to recognize in a 60-millisecond window, whereas high-frequency events may alias and appear as a pattern different from their true nature (Figure 9.8). Thus, frequency was an inherent attribute used in the training of the neural network.

Aside from the three above-mentioned scenarios affecting the types of interference patterns present, there are other influences. Because amplitude values are being used as the inputs for training the neural network, their variations throughout the shot record become important. Since no gain has been applied to the data, shallow interference has higher amplitudes than deep interference. In a 14.0-second shot record, relative amplitude values for the interference range from 20 to 0.02. It is important not to train just on the high-amplitude interference and consequently bias the network's learning. The network must be trained on a robust set of interference patterns including those with a variety of amplitude values taken from varying depths within the shot record.

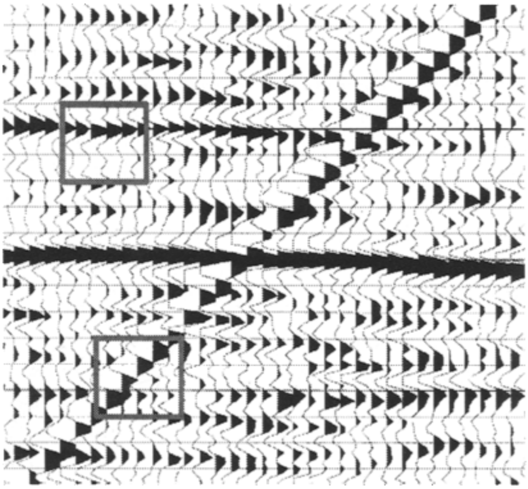


Figure 9.6. Shot display with windows of extracted data used for training the neural network. NMO correction applied. Note the curvature of the prominent multiple near the center of the display.

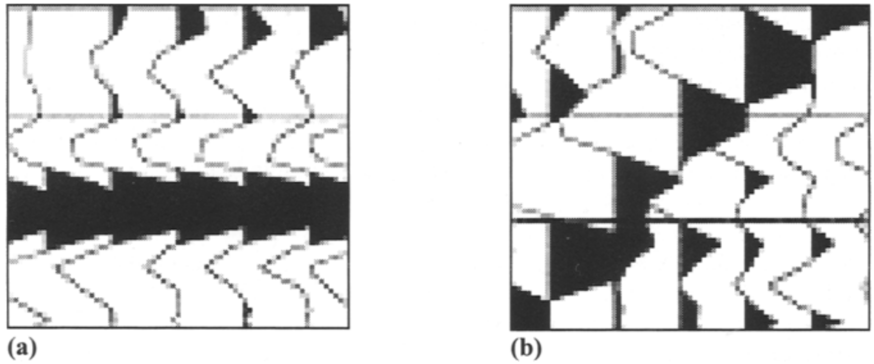


Figure 9.7. Extracted windows of data. (a) shows horizontal pattern of signal and (b) shows dipping pattern of interference.

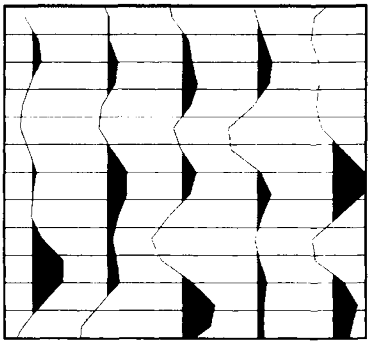


Figure 9.8. Extracted window showing spatial aliasing. This high-frequency interference pattern appears to be horizontal, similar to signal, but a larger display shows that it has a negative dip.

Closely tied to the amplitude values is the frequency of the events they represent. While frequency is not explicitly presented as part of the training set, the neural network appears to extract it from the windowed events and use it in its decision-making. Since the crew noise interference travels through the water, it behaves like a guided wave and is dispersive in nature (Yilmaz, 1987). This creates a separation of different frequencies travelling with slightly varying velocities, resulting in different arrival times in the shot record. This dispersive effect is more significant in shallow water and needs to be taken into consideration when training the network. Examples of interference with varying frequency content must be used for training to avoid biasing the network.

An extreme example of low-frequency interference occurs when it arrives in the shallow, far-offset region of the shot record. Due to NMO correction, waveforms in this region become stretched and their frequency content is lowered. In an attempt to retain as much data as possible during training, a stretch mute is not applied here. Therefore, the network must be trained to recognize this type of interference, even though it may be so severely stretched that it barely fits in the 60-millisecond window (Figure 9.9).

It should be noted that for all the different ways that crew noise interference can appear in a shot display, the neural network is trained to identify it only as a single class, interference. Whether it is rear-end, front-end, sideswipe, high-amplitude, low-amplitude, high frequency, or low-frequency interference, the network is trained to classify it as interference and output a value of −1. This is done to avoid making the classification problem more complex than necessary. Though this method seems to work well, additional classes of interference can be added to the training set.

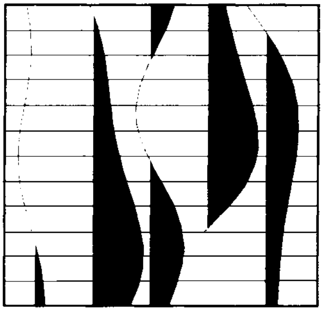


Figure 9.9. Example of low-frequency interference. Stretching of the waveform results from NMO correction and is most apparent in shallow, far-offset regions of the shot display.

2.2. Selection of signal training patterns

Patterns representing true signal have fewer variations and are therefore easier to train the network to identify. Following NMO correction, most of the signal from reflections due to subsurface geology appear to flatten and have little or no dip. This is true for most of the marine data from surveys in the Gulf of Mexico. However, geology can be complex with a range of dips from horizontal to vertical, indicating important structural features such as diapirs or faults. For this example, however, signal is considered to be any event that appears horizontal or near horizontal following NMO correction.

As with the interference, amplitude ranges were a factor for correct training of signal. Signal was found to have a similar range of absolute amplitudes between 20 and 0.02. Patterns of signal that were extracted from shallow regions of the shot record had higher amplitudes than those from greater depths. Therefore, it was important to train on a variety of patterns from all depths that covered the entire amplitude range.

Frequency also played an important role, though again it was not explicitly presented to the network during training. Due to the effects of NMO correction, patterns of signal became stretched as they approached the far offset. This is a particular problem for relatively short travel times and creates an artificially lower frequency content for signal at the far offset, while patterns closer to the near offset have higher frequencies. To train the network to recognize all types of signal, patterns with a variety of frequencies must be included in the training set.

A critical decision had to be made concerning the categorization of events as either dipping signal or interference. Sideswipe interference can approach horizontal and blend in with true signal as seen previously (Figure 9.5). Likewise, signal can dip slightly and be confused with interference. The distinction is subtle, requiring a careful approach during training. Training on signal included a majority of horizontal patterns and a few patterns with very slight dip. Signal patterns with a significant amount of dip were not included in the training set, even if they were obviously not a part of the crew noise interference. They were simply excluded from training altogether. It was decided that it was better to let the network apply its decision functions to ambiguous patterns during testing than to confuse it by training on signal and interference patterns that were too similar. This meant that some gently sloping interference might not be identified and some sloping signal may be incorrectly classified as interference.

During training, it was discovered that the neural network learned the signal patterns much more readily than it did those of the interference. This is not surprising, since there is only one way signal can appear, while interference can be either rear-end, front-end, or sideswipe. In fact, it appears that the network learned to identify patterns that were signal as one class and patterns that were not signal as the second class. While learning something about the interference, it seems more likely that the network learned to identify it by recognizing it as *not* fitting into the signal class. This is supported by the statistical composition of the training set. 75% of the 1,570 training patterns were signal, while only 25% were interference. This imbalance was by design and provided the best results during testing.

As pointed out in Chapters 2 and 4, neural networks do not process information in the same manner as human interpreters. Any person looking at Figure 9.2 sees the interference pattern as a dipping pattern even in the small training window. The network, however, only receives information about the 75 amplitude values in the small window and has no information about their spatial relationship. Even though I chose training windows based on dipping patterns, the neural network only knows that there is some complex relationship among the input PEs that distinguishes signal from interference.

3. TESTING

Following training, the network was then tested on an entire shot record. Using a sliding window that was the same size as the training window, five traces by 60 milliseconds, data were extracted from the shot record and presented to the network for classification. Similar to the training phase, 75 amplitude values for each window were input to the network and fed forward towards the output space. Based upon its training experience, the network classified the pattern contained within the window as either signal (1) or interference (-1) and output the result to a file.

Testing began at the near offset of the shot record. Once this window was extracted, presented to the network, and classified, the window was then indexed 20 milliseconds further down the shot record, overlapping with the previous window by 40 milliseconds. The new window was classified and then indexed 20 milliseconds more, overlapping the first window by 20 milliseconds and the second window by 40 milliseconds. This procedure was repeated, indexing the window 20 milliseconds further every time, creating a vertical overlap. Once at the bottom of the shot record, the window returned to the top and moved two traces towards the far offset and repeated the procedure, creating a lateral overlap of three traces. At every increment of the window, the neural network classified the pattern contained within it as either signal or interference. The sliding window was indexed vertically and laterally until the neural net classified the entire shot record.

Using the geometry of the overlapping windows, smaller windows measuring two traces by 20 milliseconds were created and used for finer discrimination of the interference. Each five-trace by 60-millisecond window was divided into six smaller windows measuring two traces by 20 milliseconds (Figure 9.10). The fifth trace of the original window was dropped for this overlapping process. If the larger window was classified as interference, then each of the six smaller windows that it was composed of was given the same classification. Due to the sliding window and the subsequent overlap, these smaller windows were each classified by the network six times. By classifying each of the smaller windows (two traces by 20 ms) a total of six times, a confidence level could be established for more accurate results. The final output was the tabulation of how many times each smaller window had been classified as interference. If the window was classified as interference more times than the confidence level, generally set to four out of six, then the window was identified as interference and its coordinates were recorded and used for later editing.

Thus, the overlapping window tabulation allowed the interference discrimination to increase in fineness from the original five-trace by 60-millisecond window to a smaller two trace by 20-millisecond window for a final output. The recorded coordinates of the small windows identified as interference were used to regenerate an edited shot display with the interference removed and replaced by zeroes.

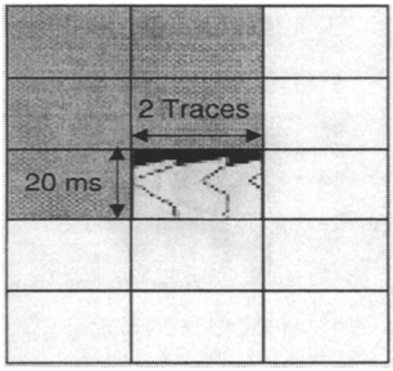


Figure 9.10. Smaller window in center is created by the lateral and vertical overlapping of the sliding larger window during testing. Original window size (5 traces by 60 ms) is shown in gray. The smaller center window is tested a total of six times by sliding the original window in two trace and 20 millisecond increments.

4. ANALYSIS OF TRAINING AND TESTING

Several criteria were used to judge the various network and training set combinations. The first criterion was the amount of correct removal of interference patterns and their continuity throughout the shot record. Since the swaths of interference are continuous events across the shot record, they should be removed with the same continuity or with as few gaps as possible. The second criterion was the amount of mistaken removal of non-interference patterns. Though the network may have identified truly dipping events for removal, they may not have been part of the crew noise interference (Figure 9.11). Hence, it was undesirable for them to be removed. The third criterion was the amount of multiple energy removed. While it was beneficial to remove some of the stronger multiples, it became detrimental to remove all of the weaker multiples that dominated the lower half of the shot records (Figure 9.12). The final criterion was robustness, a subjective measure of the network's versatility in a variety of situations. With so many different types of interference, it was important to find a network that worked well on all of them. It was discovered that sideswipe interference could rarely be detected in its entirety, but that some networks could remove the slightly dipping flanks. The best network was the one that showed the greatest amount of robustness and consistency.

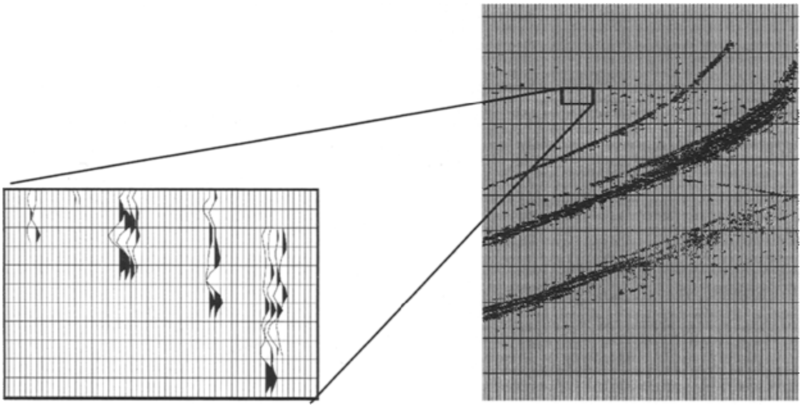


Figure 9.11. Difference display showing the mistaken removal of non-interference. These small, non-continuous dipping patterns of signal were identified as interference and removed.

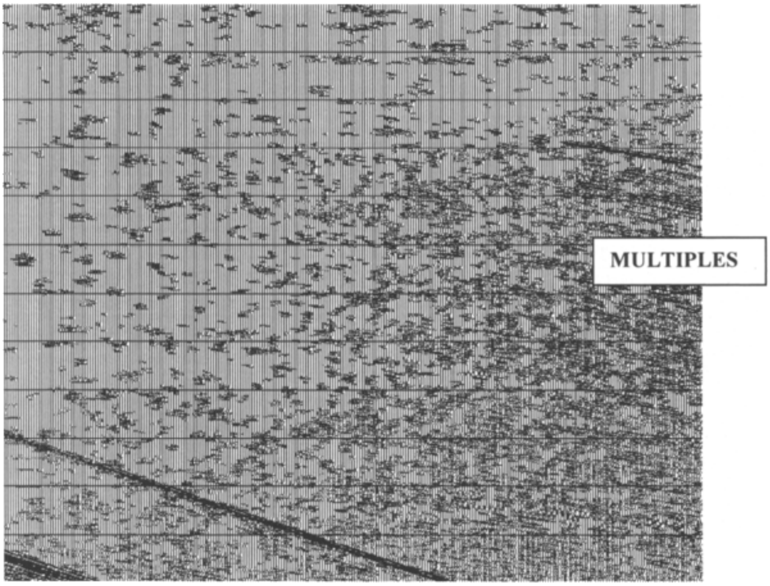


Figure 9.12. Difference display showing the over-removal of multiples at depth. Though multiples dip significantly at the far offset, excessive removal of the weaker patterns that dominate the lower portion of the shot record is undesirable.

4.1. Sensitivity to class distribution

Training started with a small set of patterns followed by testing. Patterns identified as interference were removed, and the shot display was regenerated. After examining the edited shot display and analyzing the test results, the training set was updated to correct for previous shortcomings. Again, the network was trained and tested and its results analyzed. This procedure was repeated several times until the testing results showed that a maximum amount of interference and a minimum amount of signal were removed. Common problems that were identified during this repetitive process included inadequate representation of signal and interference patterns, frequency biasing, and amplitude scaling.

When the training set included a small number of patterns selected only from one portion of one shot record, the network could train to 100% accuracy on the given patterns but the test results were poor (Figure 9.13). The training set was inadvertently biased by only using examples of high amplitude interference so the network classified any pattern with high amplitudes as interference. This was corrected by adding patterns of interference and signal with a broad range of amplitudes to the training set.

In an attempt to reduce the amount of signal being incorrectly removed, additional patterns of signal from shallow regions of the shot record were added to the training set. This contributed to an over-representation of the types of signal patterns from this region. Subsequent testing showed that the newly trained network was now removing nearly every pattern that occurred at depth (Figure 9.14). The additional training had caused the network to learn that only horizontal patterns from shallow regions of the shot record, where

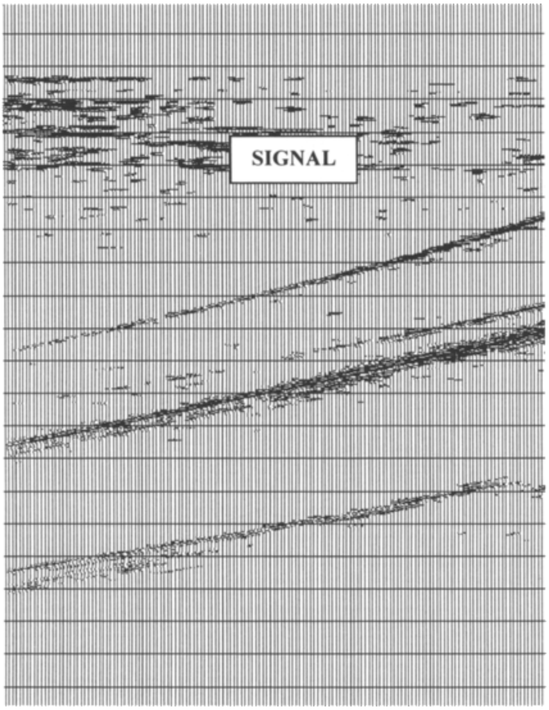


Figure 9.13. Difference plot showing data that were removed due to over-representation of high amplitude interference patterns. Even though the signal patterns are horizontal, insufficient training caused the network to incorrectly remove them because of their high amplitudes.

amplitudes were high, were classified as signal. This was corrected by adding low-amplitude patterns of signal to the training set, resulting in a more appropriate removal of interference at all depths.

Poor test results also occurred when the training set included too many examples of low-frequency signal patterns that were commonly found in the shallow, far-offset region of the shot record. Training on these patterns created a frequency bias with mixed results. It allowed the network to recognize signal in the far-offset regions of the shot record, but caused it to miss nearby patterns of interference. This was resolved by training the network on low-frequency patterns of interference from the same region.

Testing was the only method that determined the accuracy of the trained neural network. During training, it became an integral part of the process, allowing the changes in training sets and network parameters to be analyzed. A shot record or a smaller subset had to be tested every time a change was made to the training set or network architecture. The accuracy of testing for an entire shot record was determined by visual inspection.

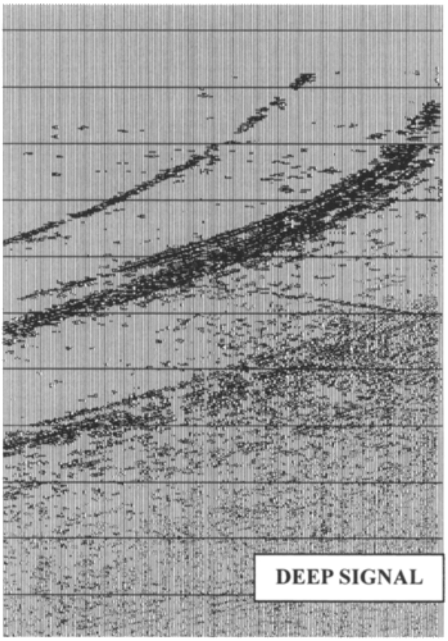


Figure 9.14. Difference plot showing data removal after over-training on shallow patterns of signal. The network learned that deep patterns were interference regardless of dip and incorrectly removed deep, low-amplitude signal.

4.2. Sensitivity to network architecture

The network judged best had 13 hidden PEs and was trained on a data set with 1,190 patterns of signal and 380 patterns of interference. This network was chosen as the best due to its moderate removal of both interference and signal, as seen in the difference plot shown in Figure 9.15. It removed nearly all of the interference without any significant gaps, except where interference was too temporally short to be detected. False removals of non-interference windows were a mere 171 out of 72,000, or just 0.24 % of the data for the entire shot record. Only a few strong multiples were removed. When tested on all nine shot records, the network performed well on all types of interference. It worked equally well for front-end and rear-end interference, and it was able to remove the greatest number of slightly dipping patterns associated with sideswipe interference (Figure 9.16).

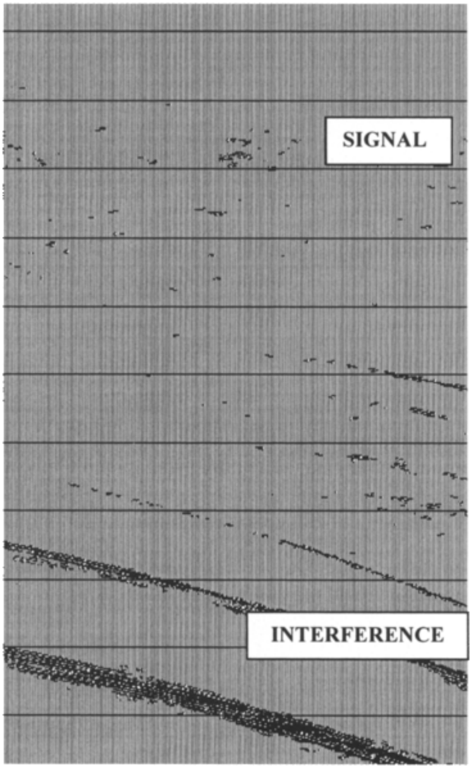


Figure 9.15. Difference display for the most effective network. It showed a thorough removal of front-end interference while only falsely removing a small number of patterns related to signal and multiples.

An alternative network that had 10 hidden PEs and was trained on fewer signal patterns removed more multiples deeper in the shot record (Figure 9.17). While I initially rejected this network for removing too much data, I will show in Section 5 that it in fact was very useful for removing noise prior to CMP stacking.

To test the effect of network architecture on the results I used the same training set, with 1,190 examples of signal and 380 examples of interference, to train networks with the same architectures and training parameters but differing in the number of hidden PEs. As seen in Figure 9.18 there is very little difference between the results when the network has 10 hidden PEs or 20 hidden PEs. The other examples in this chapter show a much greater dependence on training set design than network design.

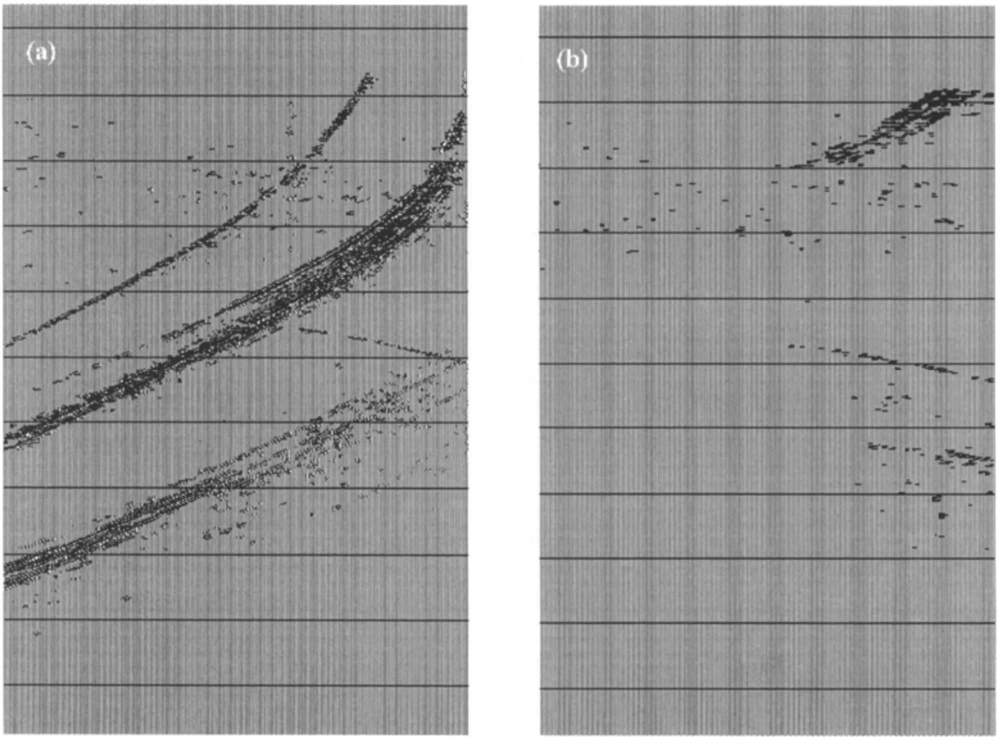


Figure 9.16. Difference displays for (a) rear-end interference and (b) sideswipe interference. Sideswipe interference was problematic for all networks.

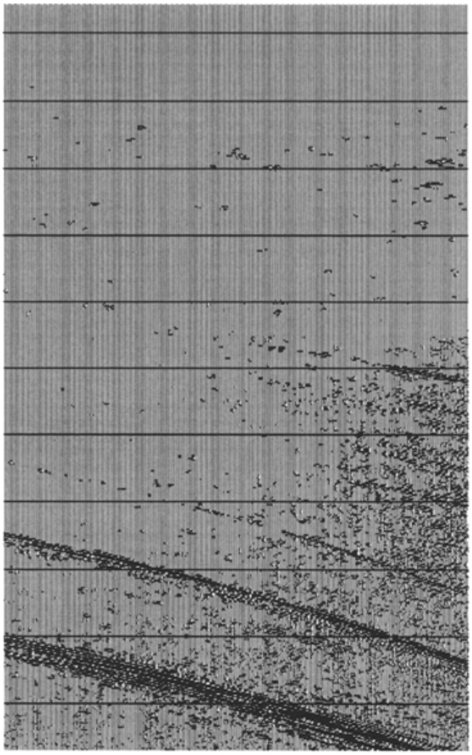
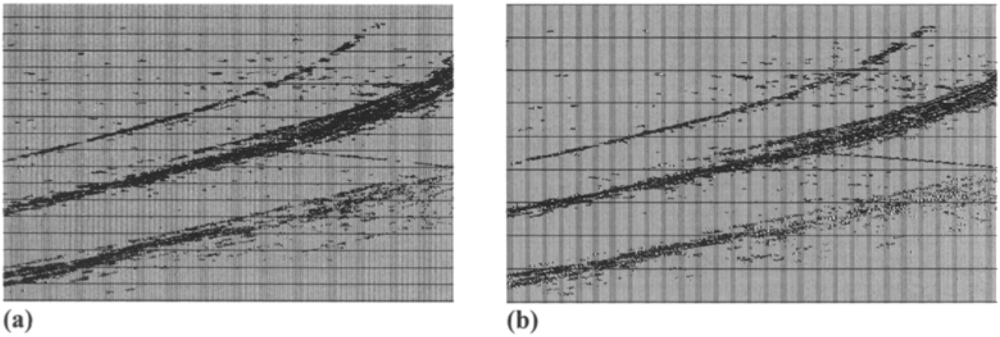


Figure 9.17. Difference display for the network with 10 hidden PEs and fewer training examples of signal. More data were removed from the deeper regions of the shot record due to the network identifying weak multiples as interference.



(a) (b)

Figure 9.18. (a) The network with 10 hidden PEs removes the same data as the network with 20 hidden PEs (b).

4.3. Effect of confidence level during overlapping window tabulation

The process of transforming the neural network's outputs of five-trace by 60-millisecond windows to smaller two traces by 20-millisecond windows utilizes a confidence level parameter. During the overlapping window tabulation, a confidence level of four was commonly used. This means that a two trace by 20-millisecond window will only be tagged for removal if it has been classified as interference by the network at least four out of six possible times. Any number less than four and the window is considered to be signal and is not removed. This confidence level allows the user to restrict how much data is removed after the network has completed its testing. Therefore, adjusting the confidence level can significantly affect the final outcome of the entire procedure.

For a trial shot record, the confidence level was decreased to three out of six. The results of the overlapping window tabulation showed that more data were removed (Figure 9.19). While the amount of interference that was removed increased, so did the number of multiple and non-interference patterns. If the additional removal of non-interference patterns is considered acceptable, then using a confidence level of three provided a more complete removal of interference.

For the same shot record, the confidence level was increased to five out of six. As expected, the overlapping window tabulation results confirmed that slightly less interference was removed along with a lesser number of multiple and non-interference patterns (Figure 9.20). If the decreased removal of interference was within acceptable limits, then this confidence level allowed for the least amount of non-interference to be removed.

The desired amount of pattern removal dictates which confidence level is optimal. If the goal is to remove as much interference as possible, then a confidence level of three is best. Consequently, more non-interference patterns will be removed. If the goal is to reduce the number of non-interference patterns removed, then a confidence level of five works best. The tradeoff is a decreased removal of interference patterns.

4.4. Effect of NMO correction

Accurate NMO correction is critical for the correct identification of crew noise interference. Since the network recognizes patterns based upon their dip or lack of dip, it is important that patterns related to signal are correctly returned to their horizontal positions while the interference remains dipping. Regional velocities must be known reasonably well for the survey location in order to be used for the NMO correction prior to neural network testing. In some cases though, regional velocities may not be accurate or may not fit complex local geology. The result of inaccurate NMO correction is a shot record with geologic horizons dipping falsely. It is therefore important to determine how accurate the NMO correction must be so that the network performs properly.

Velocities for the nine shot records used for testing were taken from known velocity functions for the region of the survey. NMO correction was applied to each of the shot records using these velocities. For the purposes of this study, this was considered to be the optimal NMO correction, restoring geologic boundaries to their true positions. This is the standard to which all other results were compared.

To study the effects of NMO under correction, each of the velocities in the function was increased by 5, 10, and 20% in three separate tests. Following inaccurate NMO correction, the shot record was tested by the neural network. The actual interference was removed

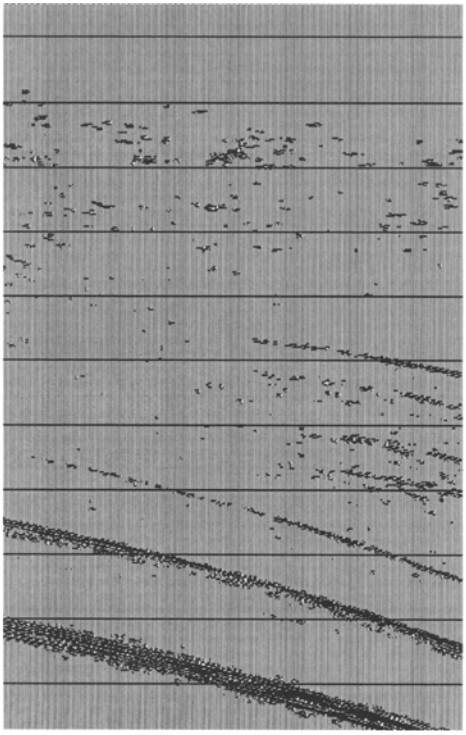


Figure 9.19. Difference display for confidence level = 3. More interference and signal was removed.

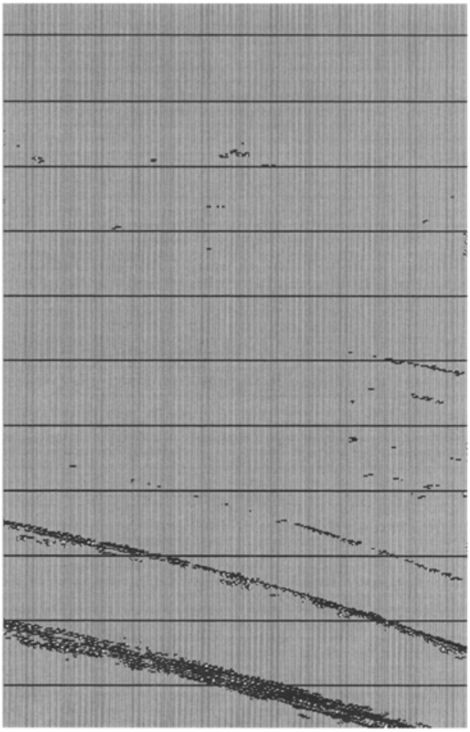


Figure 9.20. Difference display for confidence level = 5. Fewer data were removed creating less continuity of interference removal.

similarly in all three cases, showing little degradation. At 5% under correction, only a slight amount of signal was falsely identified as interference. At 10% and 20% under correction, a significant amount of signal was removed beyond acceptable levels. This occurred primarily in the shallow, far-offset region of the shot record, where under correction caused events to dip downwards.

The effects of NMO over correction were studied in a similar fashion by decreasing the velocities by 5, 10, and 20%. Over correction caused shallow events in the far offset to dip upward. At 5% over correction, a small amount of signal was incorrectly removed. At 10% and 20% over correction, much more signal was removed. Over correction caused more false removal of signal than similar levels of under correction and should therefore be avoided. Figure 9.21 shows the effects of NMO under correction and over correction. The false removal of signal at shallow depths towards the far offset is insignificant due to standard muting practices that typically remove this region prior to stacking. Therefore, it appears that the amount of acceptable error in the velocity function is conservatively estimated to be approximately ± 10% of the correct values. Within this range, the neural network performs well and removes the appropriate interference while only removing a small number of patterns related to signal.

5. VALIDATION

After neural network classification, the edited shot records were passed through a standard processing flow (Table 9.2) using parameters identical to those of other Gulf of Mexico surveys. The effects of deconvolution, common midpoint (CMP) stacking, and other processes were studied. Two separate processing flows for the same line were created for comparison. Both were identical with the exception of the inclusion of the neural network classification program in the second flow. At every step, the effect of the removal of the interference was compared to the results achieved using the standard flow without the neural network program. The analysis identified strengths and weakness for the implementation of the neural network program. All processing was done using Geovecteur ® by CGG, Inc.

Table 9.2
Processing flow for incorporation of neural network crew noise removal

Steps
Read in raw shot displays from ship in SEG-Y format
Apply neural network
Apply deconvolution and produce a brute CMP stack
Apply dip moveout (DMO) and produce a stack
Apply migration and produce a stack
Apply filters and multiple attenuation methods
Produce the final stack

5.1. Effect on deconvolution

Deconvolution is often applied to seismic data to improve its resolution by compressing the seismic wavelet (Yilmaz, 1987). A major concern about the use of the neural network and the subsequent zeroing of interference centered about the effect the zeroed amplitudes would have on the deconvolution process. Typically in seismic data, an inverse filter is convolved with the seismic wavelet to produce a spike that more accurately represents the earth's impulse response. If, however, a portion of the seismic wavelet was zeroed, its convolution with the inverse filter would be altered and perhaps cause gaps to still remain.

Geovecteur's single-channel deconvolution program provided a simple solution to the problem. In areas where gaps occurred due to the removal of interference, the filter response was used to fill in the voids. As long as the gap was smaller than the filter operator length, the gaps were successfully filled during deconvolution. With typical filter operator lengths of 250 ms, it was rare for gaps in the data to exceed this and remain unfilled.

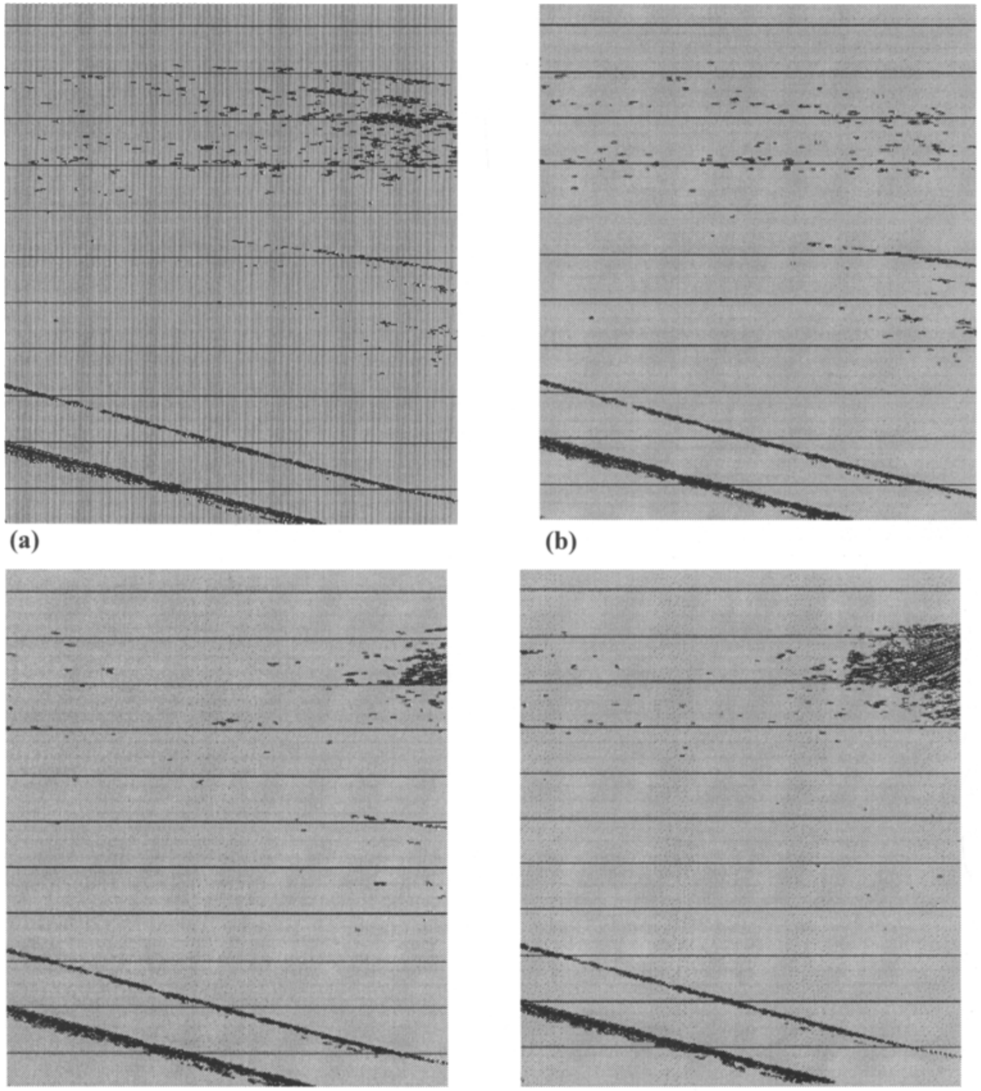


Figure 9.21. Difference displays showing the effects of NMO under correction and over correction. (a) 5% and (b) 10% under correction. (c) 5% and (d) 10% over correction. Events most affected were in the shallow, far-offset region, causing incorrect classification as interference.

5.2. Effect on CMP stacking

The neural network operates in the shot domain to identify interference and remove it. For the purpose of interpretation, shots are stacked to create entire lines. The process of stacking tends to attenuate random noise, while retaining coherent events. Left untouched, the seismic crew noise interference was indeed coherent and could still be seen following stacking. Therefore, it was important to analyze the efficiency of removal of interference in the CMP stacks.

While the network neatly removed large swaths of interference from shot records, when the edited versions were stacked, interference was still evident. Analysis of CMP gathers showed that the stacked interference was the result of low-amplitude interference found deep in the shot records. Though low-amplitude in an ungained shot record, the coherent energy of the interference was magnified during the stacking process. Hence, by applying the network to ungained shot records, this interference was overlooked and appeared in the stack (Figure 9.22).

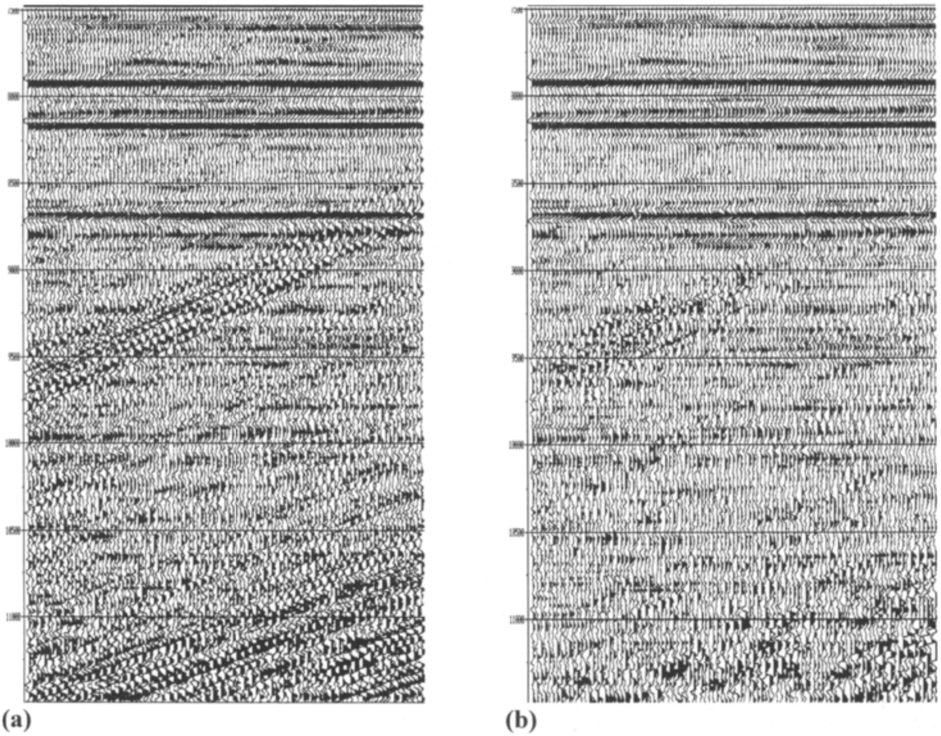


Figure 9.22. (a) Stack with no noise attenuation and (b) stack after processing with the first neural network targeting high-amplitude interference. Note that some interference remains in the stack.

The solution to this problem was obtained by applying two different neural networks to the shot records. The first network removed high-amplitude interference while the second network targeted lower amplitude interference deep in the shot record. The result was increased removal of interference over a broad range of amplitudes. When stacked, the interference was more effectively attenuated, allowing deeper events to be recognized. The combination of two different networks succeeded in removing a maximum amount of interference while retaining nearly the entire original signal (Figure 9.23). Total processing time to apply both networks to a shot record and perform the data I/O and confidence windowing was approximately 1.5 seconds.

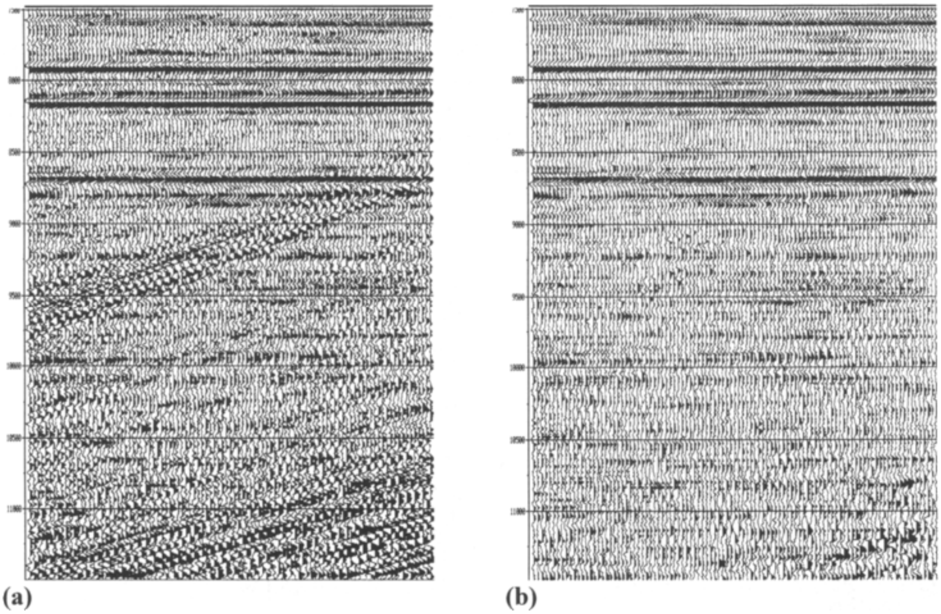


Figure 9.23. (a) Stack before and (b) stack after neural network identification and removal of interference.

6. CONCLUSIONS

In marine seismic surveys where crew noise from rival ships is present, a back-propagation neural network is capable of identifying the interference based upon the slope of its pattern in the shot domain. Furthermore, when the identified interference is removed and edited shot records are generated a standard processing flow can be applied to the data to produce a significantly improved CMP stack. In the case of AVO analysis, the zeroed interference is simply flagged and not included in the analysis, ensuring more accurate results. In every case, the network simply identified the coordinates of the interference, allowing post processing to remove it or replace it at the processor's discretion. Because of the large number of shot records involved in each CMP stack, a network could remove a substantial portion of the data deep in the shot record without harming the stack. Training samples were drawn only from four shot records but the network results were robust enough to apply to the entire survey area.

Two problems the network could not adequately address are sideswipe interference and dipping geologic structure. Sideswipe interference is very difficult to remove and this technique had only limited success in separating such interference from signal. While the network was not tested on any shot records that exhibited dipping geologic structures, it should be remembered that the network is not using dip as a classification feature but amplitude patterns. Hence, dipping structure may or may not pose a problem.

While the approach described in this chapter was successful for data from a deep-water survey, future work might focus on the application of gain and some amplitude normalization to make the network less site specific. This might have the added benefit of making the training set less sensitive to small changes. While there are additional parameters that could be altered to optimize the results, this study has demonstrated the feasibility of utilizing a neural network for the identification and removal of seismic crew noise interference.

REFERENCES

Buffenmyer, V., 1999, Neural network approach to seismic crew noise identification in marine surveys: MS Thesis, The University of Arizona, Tucson, AZ.

Canales, L., 1984, Random noise reduction: SEG expanded abstracts, 525-527.

Hargreaves, N., Manin, M., Gratacos, B., Micklewright, I. and Perkins, C., 1997, Seismic crew noise - A zero time-sharing solution: 61st Mtg. Eur. Assoc. Expl Geophys., Extended Abstracts, European Association Of Geophysical Exploration, Session:B040.B040.

Lynn, W., Doyle, M., Larner, K., Marschall, R., 1987, Experimental investigation of interference from other seismic crews: Geophysics, **52**, 1501.

Manin, M., and Bonnot, J., 1993, Industrial and seismic noise removal in marine processing: Abstracts of the 55th Meeting of the EAEG, B031.

Yilmaz, O., 1987, Seismic Data Processing: Society of Exploration Geophysics, Tulsa, OK.

Chapter 10
Self-Organizing Map (SOM) Network For Tracking Horizons And Classifying Seismic Traces

Lin Zhang, John Quieren, and James Schuelke

1. INTRODUCTION

In the analysis and interpretation of seismic data, event picking and evaluation of the different seismic responses are very important, though often difficult and time consuming. The events on a seismic section usually appear consistently from trace to trace, and interpreters utilize these events to construct geological and geophysical models. The interpreters normally assess not only reflection amplitude, but also the morphology or character of the reflection, i.e., wavelet shape. Tracking a horizon can be automated by using a neural network to classify the wavelet morphology. Horizon picking, however, is commonly done by hand. Attempts to automate the process have been hindered by absence of a clear, robust and universal picking algorithm.

Horizon tracking and facies mapping applications were briefly described in Chapter 8. The idea of using a neural network to automate horizon tracking dates to Huang (1990) and Harrigan and Durrani (1991). In this chapter, we will explore in more detail how to use a Self Organizing Map (SOM), to track horizons and then classify waveforms in the vicinity of the horizons. Tracking a particular seismic event (horizon) in 3D defines a surface, and the self-organizing map network can be applied to the seismic traces proximal to this surface, resulting in a classification of the seismic responses into geological features. The classified waveforms can show patterns related to geologic properties in the absence of well log data. With well log or other ancillary data, the classified waveforms can be correlated with properties such as porosity or shale volume.

In this chapter we will examine the use of the self-organizing map to track horizons and classify seismic traces from data collected in the Blackfoot field near Calgary, Alberta[*].

2. SELF-ORGANIZING MAP NETWORK

The SOM was described in Chapter 5. The basic architecture of the two-dimensional self-organizing map network is shown in Figure 10.1. The PEs in the Kohonen layer are located in either a one- or a two-dimensional lattice and fully connected with the input pattern. Alternatively, the PEs may be placed at random locations within a two-dimensional grid. The Kohonen neurons compete among themselves to be activated. Only one output neuron, which has the minimum distance between itself and the input pattern using the Euclidean distance measurement, is activated. During training, the Kohonen PE with the smallest distance from the input pattern updates its weights to be closer to the same input pattern. Within a neighborhood around the winning PE, additional PEs also update their weights. The fact that neighboring PEs also adjust their weights to become more similar to the training pattern allows the PEs in the Kohonen layer to become topologically ordered, i.e., patterns that are spatially close activate PEs that are spatially close.

---

[*] Data supplied by the CREWES project, University of Calgary, Canada.

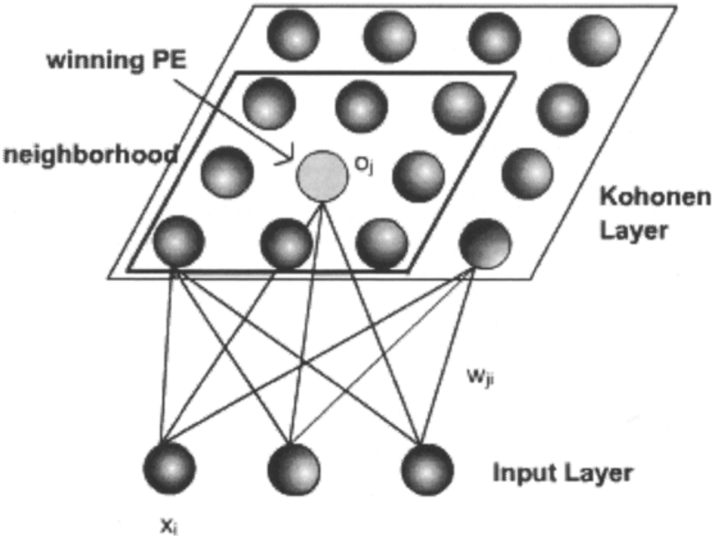


Figure 10.1. The architecture of the self-organizing map (SOM) network

Advantages of the SOM are:

1. The SOM learns to categorize input vectors. It also learns to distribute the Kohonen neurons in the two dimensional network based on the density of input patterns. In other words, the Kohonen two dimensional feature map allows more neurons to recognize parts of the input space where many input patterns are presented and allocate fewer neurons to parts of the input space where few input patterns occur.
2. The output of the SOM reflects the topology of the input patterns. The neurons next to each other in the Kohonen layer learn to respond to similar input patterns.
3. Since the SOM allows the winning PE's neighborhood to update weights, the transition of output vectors (weight vectors) is much smoother than competitive neural networks, which only update the winning neuron.

3. HORIZON TRACKING

3.1. Training Set

In the Blackfoot field, southeast of Calgary, Alberta, a 3D multi-component seismic survey was committed in October 1995 by Boyn Exploration Consultants Ltd. for the CREWES project of the University of Calgary. The objective of the survey was to evaluate possible improvements to hydrocarbon exploration in this region from utilization of the 3D, multi-component data. Oil and gas in the Blackfoot area are produced from the Glauconitic formation of the upper Mannville group. Three horizons: top-Glauconitic (*glctop*), lower-Mannville (*lmann*) and Mannville (*mann*) are available from previous geological interpretations.

To explore for more detailed information from the seismic data set and to investigate performing a quick and accurate horizon tracking, the self-organizing map architecture is investigated.

Based on the self-organizing map network, an algorithm is derived to track horizons across seismic traces. The major advantage of the self-organizing map network is to learn the topology of the input pattern; thus, the inputs to the network are the seismic traces with a time window of a specific size. The basic idea is to predict the input trace sample by sample. Each sampling point in the seismic traces, plus five samples above it and five samples below it, are extracted as a training pattern. The window length of each training pattern is 22 ms due to the sampling rate of 2 ms. To give more geological and geometrical information to the network, using this trace as center, a certain number of traces to the left and the same number of traces to the right have been grouped together to create a new training pattern. Several training sets are created and evaluated independently. Table 10.1 shows the parameters in each training set. Hence, training set 3 includes two traces to the left of the sampled trace and two traces to the right for a total of 5 traces.

To create the training patterns, a sliding window was used along each trace. The window was 22 ms in length and was moved 2 ms at a time. Hence, between a time interval of 900 and 1200 ms, we had a total of 150 training patterns for each of the 81 crosslines at one particular inline location. The only difference between the training sets in Table 10.1 is whether adjacent traces were concatenated to the training pattern for each time window.

A two-dimensional Kohonen layer was used with the number of PEs equal to the desired number of output classes. The output classes are the different morphologies of the wavelets. The network was trained for 20,000 iterations. A step size of 0.9 was used initially and then decreased by 0.02 after every 1,000 iterations. A random topology was used to place the PEs at the start of training. We experimented with class sizes of 10, 20 and 50. After training the Euclidean distance between the class weight vector and a training trace was computed. When 50 classes were used, we calculated the lowest overall error for the data shown in Figure 10.2. The difference in error, however, between 10 and 50 classes was very small so we chose to complete the analysis with 10 classes.

Table10.1
Parameters for different training sets

	Training set 1	Training set 2	Training set 3	Training set 4
Number of traces	1	3	5	9
Samples in each trace	11	11	11	11
Input vector dimension	11	33	55	99

3.2. Results

The three horizons we want to map: top-Glauconitic (*glctop*), lower-Mannville (*lmann*) and Mannville (*mann*), occurr between 1000-1100 ms. Figure 10.2 shows the raw seismic data with the three horizons at inline 85. As seen in Figure 10.2, *lmann* has a stronger response than *mann* and *glctop*. The event corresponding to horizon *mann* has less continuity between crossline 130 and 150. Horizon *glctop* has a good continuity, however, it is weak at crossline 120. The data between 1100 –1170 ms are quite fuzzy, thus, it is hard for the interpreter to trace these events. These data, however, are still included in the neural network training.

The SOM network was compared to the manual seismic horizon tracking using the three different training sets. During the competitive learning, since similar geological features would have similar waveforms, the SOM network assigned the training patterns (seismic traces) into appropriate ordered "geological" classes based on the minimum Euclidean distance. The weight vector in each class matches a group of seismic traces that have similar waveforms. Each class has an individual color (or gray level) so events can be tracked by individual or groups of colors. Figures 10.3 and 10.4 show the tracking of the horizons when one and five adjacent traces are used for training.

The use of five adjacent traces provided the best overall match to the manual selection of the three horizons of interest. Our various trials with numbers of classes, numbers of adjacent traces, and network training parameters yield nearly the same result indicating the training process was stable. At the bottom of each figure, the weight vector for each class is shown; the subtle similarity and differences in the classes can be observed. The classes assigned to each of the horizons as a function of the number of traces in the input pattern are shown in Table 10.2.

The three horizons of interest are not always uniquely separated from each other or from other horizons in the data so this technique is not useful for horizon picking. The horizons, however, do show good continuity across the data set indicating the technique can be successful for horizon tracking. Interestingly, even the region of poor quality data, between 1100 and 1200 ms, shows reasonably good continuity along horizons indicating the technique has some robustness in the presence of noise.

Table 10.2
Classification of horizons as a function of training pattern size

Number of traces	*mann*	*lmann*	*glctop*
1	9, with some class 10 at end of line	10	2
3	6	6, with some class 7 in the center of the line	2,3
5	4,5	6	10
9	6,7	7	7,8

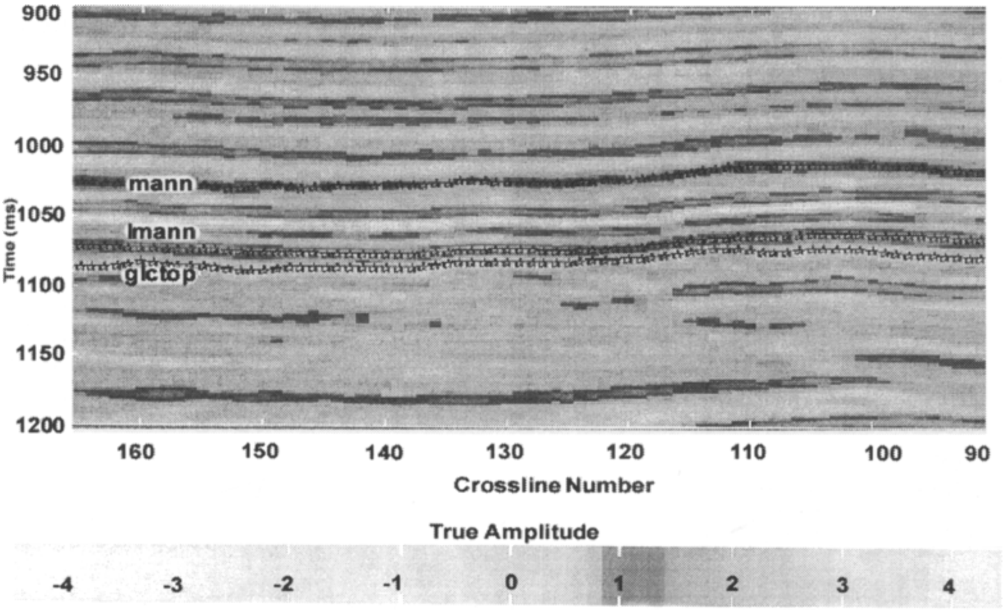


Figure 10.2. Raw seismic data for inline 85. Crosslines are 15 m apart so distance between horizontal tick marks is 150 m. The three horizons of interest *mann*, *lmann*, and *glctop* are noted with tickmarks.

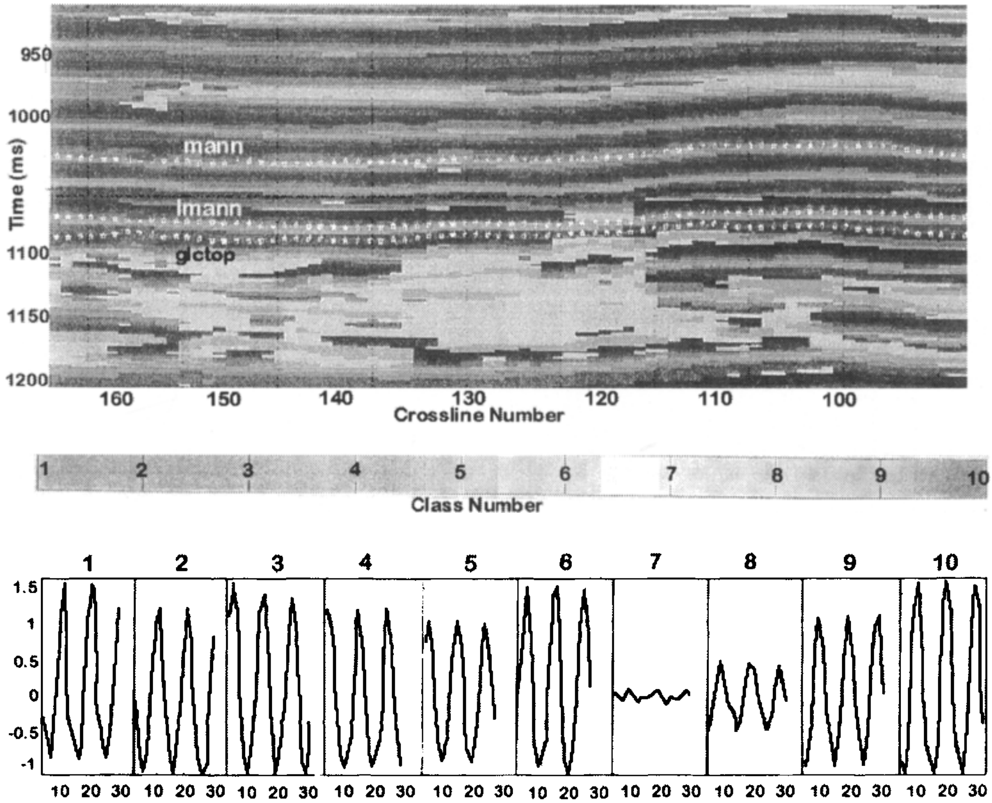


Figure 10.3. Results after using the SOM to track the three horizons. The graphs at the bottom of the figure are the SOM weight vectors for each class. Three traces were concatenated as input to the network. The network had 10 PEs, one for each of the 10 desired classes.

These results have demonstrated a relationship between the SOM classes and interpreted horizons. As such, the SOM classes can be used to assist the interpreter in the selection of horizons. Additional research is required to develop rules for automatically obtaining the horizons from the SOM classes.

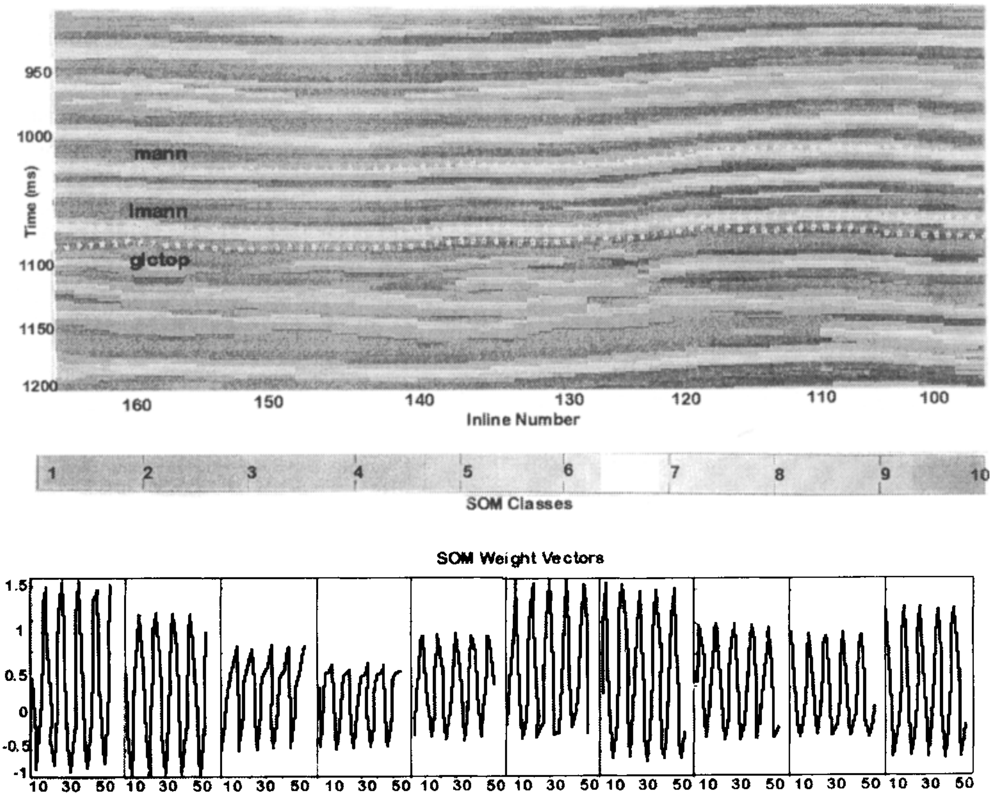


Figure 10.4. Results after using the SOM to track the three horizons. The graphs at the bottom of the figure are the SOM weight vectors for each class. Five traces were concatenated as input to the network. The network had 10 PEs, one for each of the 10 desired classes. Notice that the network classification shows more continuity between 1100 and 1200 ms when five traces are used as input.

4. CLASSIFICATION OF THE SEISMIC TRACES

Once a horizon or horizons are tracked, our next objective is to classify traces within the horizons to produce maps that can be correlated with geologic properties such as porosity. When well log data are available, a program such as Emerge® (Hampson-Russell, Inc.) can be used to convert seismic data to porosity values using a probabilistic neural network. Our goal is to compare a classification of seismic traces to calculated porosity values to establish whether unsupervised classification results can be correlated with meaningful geologic properties.

In the Blackfoot area, a total of 13 wells were tied with the seismic data and converted to time-based logs. Each well has a porosity and acoustic impedance (from compressional velocity) log. As a first step, an acoustic impedance volume was created through a model based inversion algorithm using the raw seismic volume and well data. Next, 16 attributes were calculated as listed in Table 10.3. The probabilistic neural network in the software package Emerge (developed by Mobil, Inc. and Hampson-Russell, Inc.) was applied to training data using the 16 attributes plus the acoustic impedance at each well location. Once the network learned to estimate porosity at the well locations, it could be applied to the entire data set to produce a porosity map as shown in Figure 10.5.

Table 10.3
List of seismic attributes calculated for the porosity estimation

Attributes
Amplitude envelope
Amplitude weighted cosine phase
Amplitude weighted frequency
Amplitude weighted phase
Average frequency
Apparent polarity
Cosine instantaneous phase
Derivative
Derivative instantaneous amplitude
Dominant frequency
Instantaneous frequency
Instantaneous phase
Integrate
Integrated absolute amplitude
Second derivative
Time

Two reservoirs with good porosity were found at 1055-1075 ms, which is above the lower-Mannville horizon, and 1080—1100 ms, which is below the lower-Mannville horizon. The porosity values of the 20 time samples between 1080—1100 ms were averaged and plotted in Figure 10.5. A high porosity channel, which contains five wells, 16-08, 29-08, 09-08, 08-08, 01-08, trending from north to south was predicted. Wells 13-16 and 09-17 are also located in high porosity areas. Figure 10.6 shows the raw seismic amplitude at 1088 ms with the location of thirteen wells. No channel can be seen. The acoustic impedance at 1088 ms is shown in Figure 10.7. In general, high porosity sand has low acoustic impedance. A low acoustic impedance channel is found from north to south. Wells 08-08, 09-08, 29-08 and 16-08 are included in this "impedance" channel. The well 01-08, which has very good porosity, is not included in the low impedance channel. The acoustic impedance at wells 09-17, 13-16 and 14-09 were also observed to be rather low.

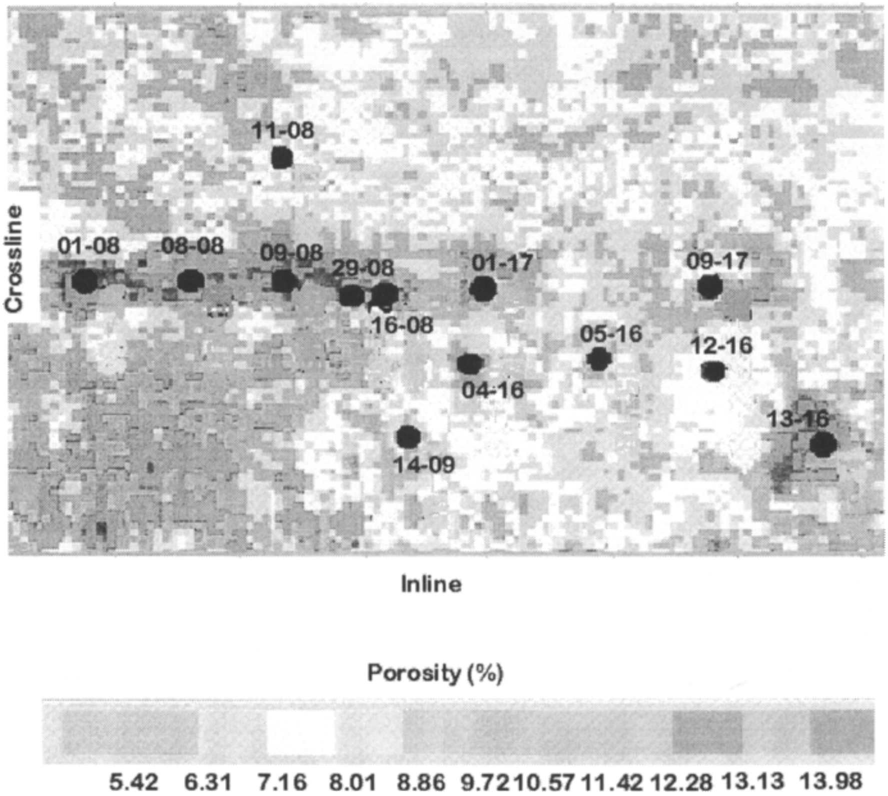

Crossline
Inline
11-08
01-08
08-08
09-08
29-08
16-08
01-17
09-17
05-16
12-16
04-16
13-16
14-09
Porosity (%)
5.42
6.31
7.16
8.01
8.86
9.72
10.57
11.42
12.28
13.13
13.98

Figure 10.5. Computed porosity at 1100 ms using a 20 ms window average.

The SOM network could be trained with traces from either the raw seismic data (meaning constant length portions of seismic traces) or the acoustic impedance data (or both). The classification would be based on similarities and differences between the shapes of the traces. Four training sets were created based on different time windows for both the seismic traces and acoustic impedance data. Table 10.4 lists the time window and sample numbers in the training sets.

Table 10.4
Input in the different training set

	Training set 1	Training set 2	Training set 3	Training set 4
Time window (ms)	1060-1100	1070-1110	1080-1120	1080-1100
Samples in each trace	20	20	20	10

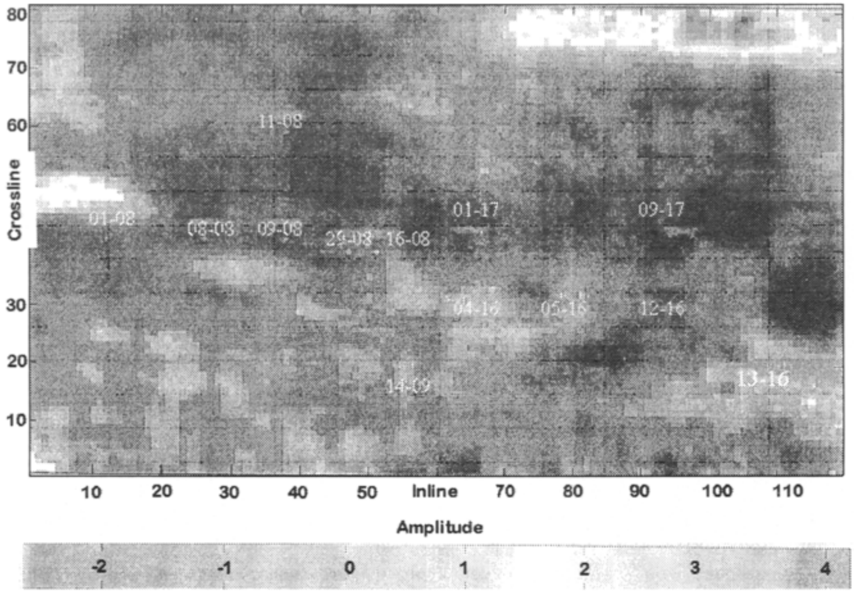

80
70
60
30
20
10
Crossline
11-08
01-08
08-08
09-08
29-08
16-08
01-17
09-17
04-16
05-16
12-16
13-16
14-09
10
20
30
40
50
Inline
70
80
90
100
110
Amplitude
-2
-1
0
1
2
3
4

Figure 10.6. Seismic amplitude data at 1088 ms. These data do not show the channel with high porosity that was identified in the porosity data.

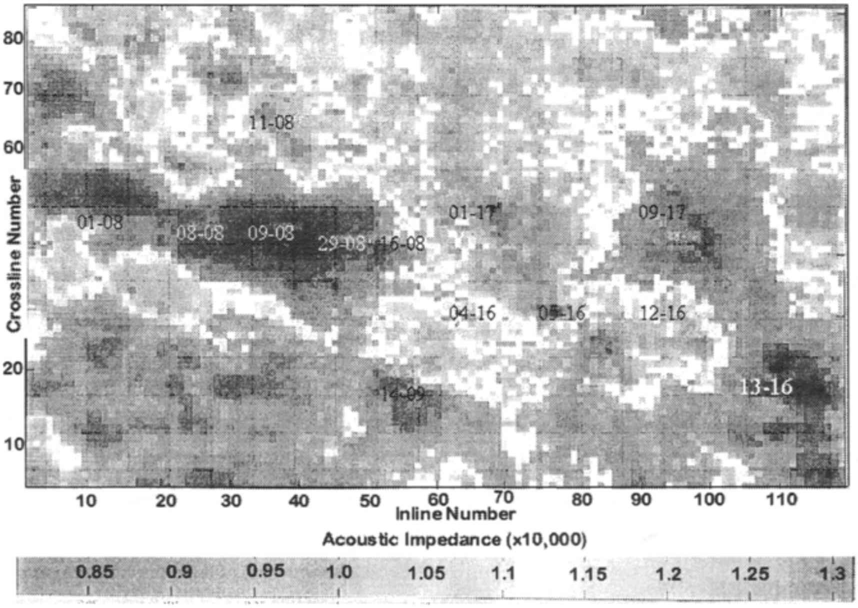

80
70
60
20
10
Crossline Number
11-08
01-08
08-08
09-08
29-08
16-08
01-17
09-17
04-16
05-16
12-16
13-16
14-09
10
20
30
40
50
60
70
80
90
100
110
Inline Number
Acoustic Impedance (x10,000)
0.85
0.9
0.95
1.0
1.05
1.1
1.15
1.2
1.25
1.3

Figure 10.7. Acoustic impedance data at 1088 ms provide more detail than amplitude data and highlight the same high porosity channel as shown in Figure 10.5.

The training set consisted of 9,639 patterns. The data set contained 81 crosslines and 119 inlines and we sampled traces at each intersection point. The SOM network contained the same number of PEs as training classes and was trained for 50,000 iterations.

Since the seismic data were assumed to represent the response of the geology encountered in these wells, the seismic waveforms were derived as the average of the nine seismic traces around each well location. The averaged seismic waveforms and acoustic impedance at the thirteen wells between 1060—1120 ms are shown in Figure 10.8 and Figure 10.9. These averaged seismic waveforms will be applied as indicators for the performance of the self-organizing map network.

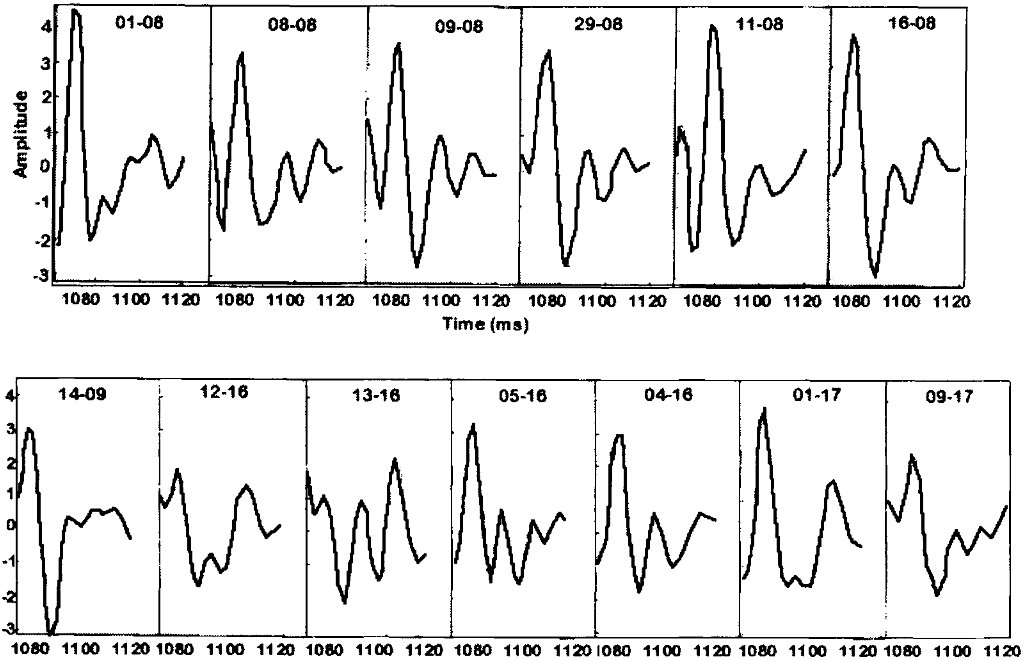


Figure 10.8. Average amplitude waveforms between 1080 and 1120 ms for each of the 13 wells in the data set.

Through the competitive learning process, the network was able to simplify the total variability of the seismic response into different classes. The statistical analogy to this process is K-means clustering. The process of training the network and selecting the classes is listed in Figure 10.10.

Since the SOM classes are only indirectly related to the geological features, we have flexibility in choosing the different window lengths and numbers of classes. In the next section we test the effect these two variables have on the overall classification results.

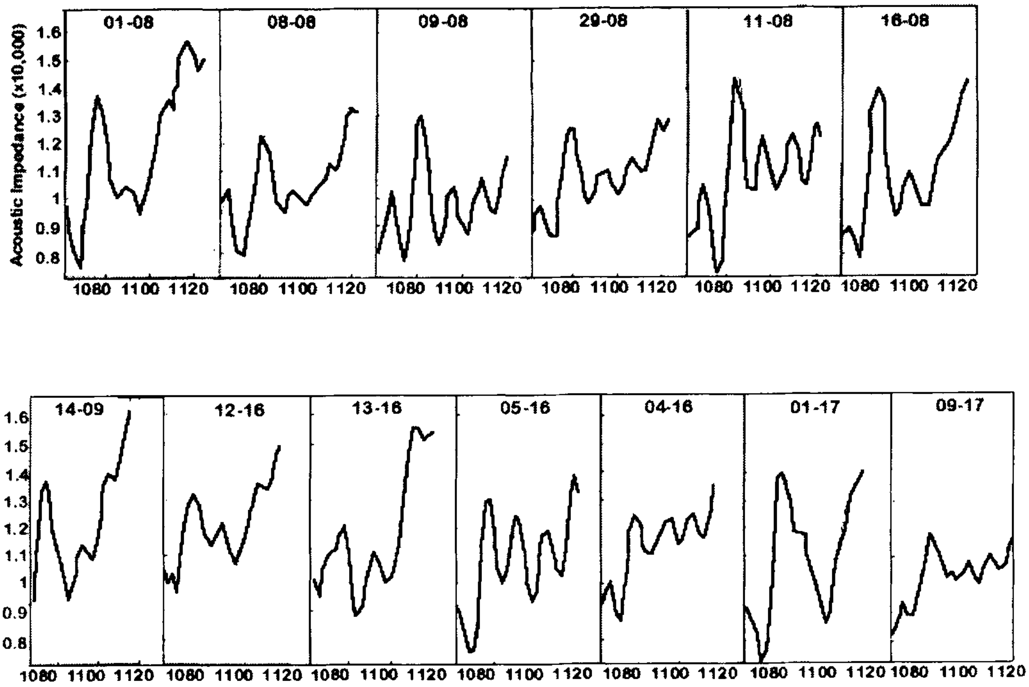


Figure 10.9. Average acoustic impedance waveforms between 1080 and 1120 ms for each of the 13 wells in the data set.

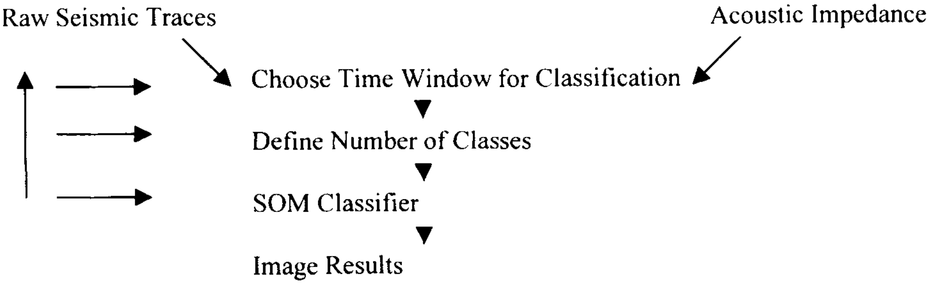

Raw Seismic Traces
Acoustic Impedance
Choose Time Window for Classification
Define Number of Classes
SOM Classifier
Image Results

Figure 10.10. Work flow to classify seismic traces with the SOM network.

4.1. Window length and placement

The first step in the analysis was to assess the variability of the seismic response in the different time windows and determine which time window best characterized the response. Ten classes were specified as the required output from the SOM classification process in each case. We tested both the raw seismic data and the acoustic impedance data.

Each time window produced different wavelets for the same classes and therefore different classification maps indicating sensitivity to changes in the time interval that is selected. The north-south channel was not detected in the classification of the raw seismic data (Figure 10.11).

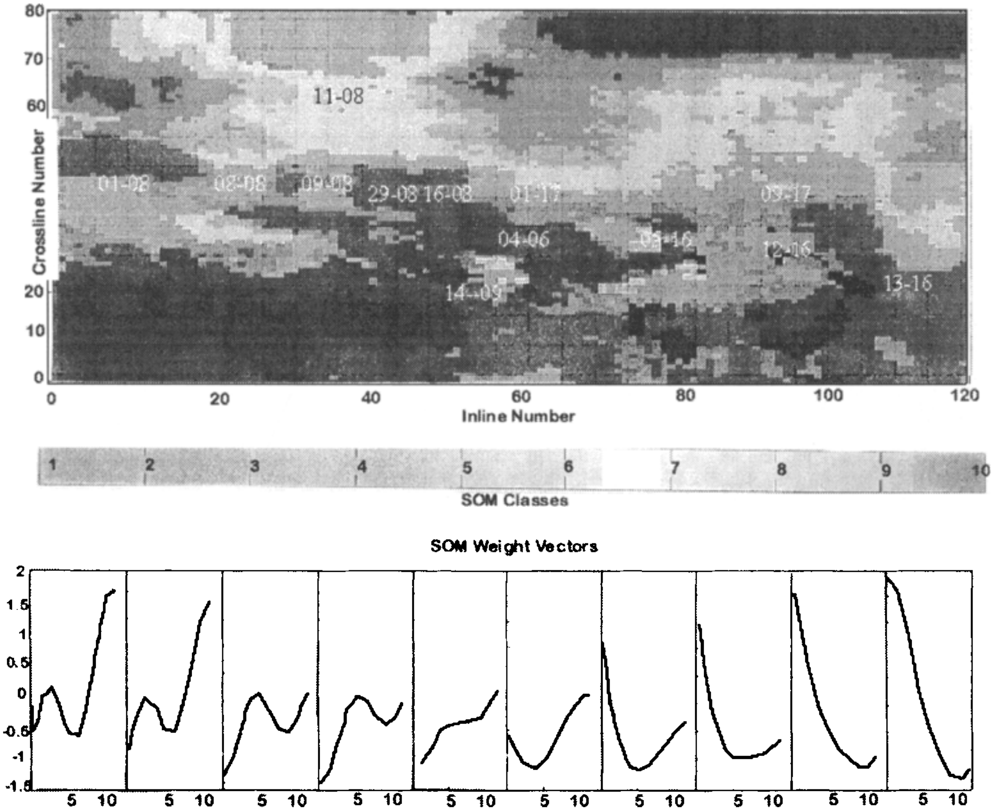

Crossline Number
80
70
60
20
10
0
11-08
01-08
08-08
09-08
29-08
16-08
01-17
09-17
04-06
08-16
12-16
14-09
13-16
0
20
40
60
80
100
120
Inline Number
1
2
3
4
5
6
7
8
9
10
SOM Classes
SOM Weight Vectors
2
1.5
1
0.5
0
-0.5
-1
-1.5
5 10

Figure 10.11. SOM classification of seismic amplitude data between 1080 and 1100 ms. The graphs at the bottom of the figure are the SOM weight vectors for each of the 10 classes.

Figure 10.12 shows the results for classification of the acoustic impedance traces. There was a bit more consistency between the wavelets assigned to each class for the different time intervals. The north-south channel was not clear at time window 1060-1100 or 1070-1110. At 1080-1120, the channel began to appear as class 3 and included wells 08-08, 09-08, 29-08, and 09-17. The waveforms of these four wells have many similarities; thus, it is reasonable to classify them together. The channel was clearest at time window 1080-1100 ms and was mapped by classes 1 and 2. The channel contains the same five wells that were predicted in the high porosity region by the previous interpretation. The averaged waveforms at the thirteen wells are shown in Figure 10.9. Note also the area around wells 09-17, 13-16 is classified as classes1 and 2. This is consistent with the previous porosity prediction provided by probabilistic neural network.

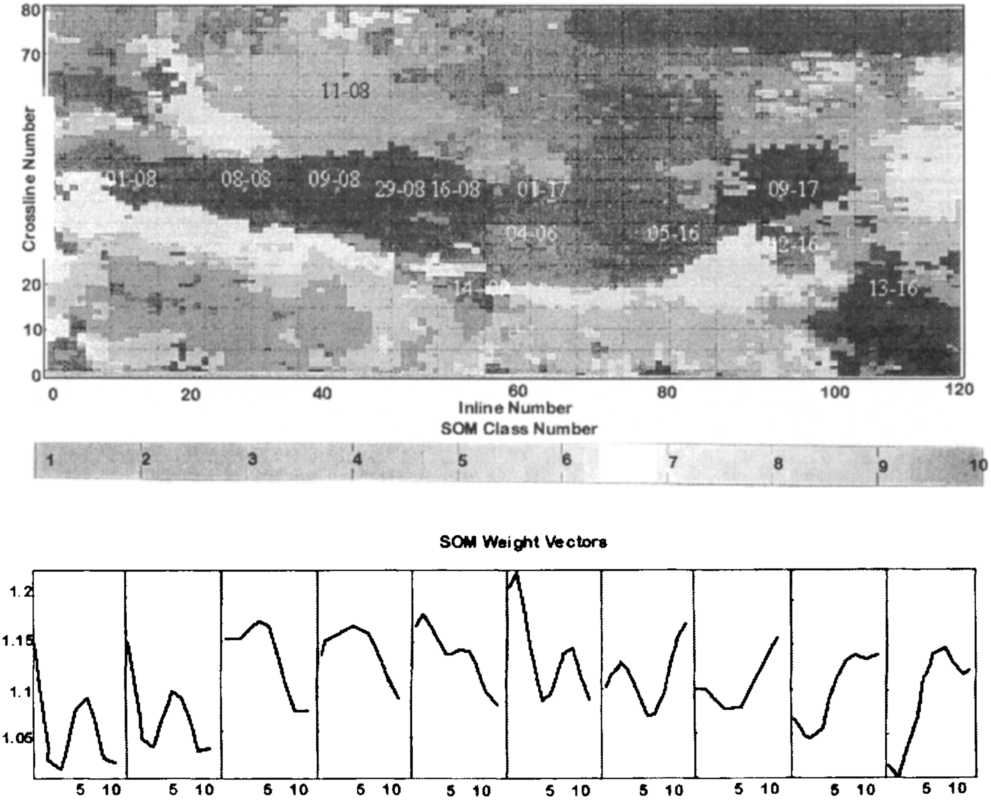


Figure 10.12. SOM classification of acoustic impedance data 1080 and 1100 ms. The graphs at the bottom of the figure are the SOM weight vectors for each of the 10 classes.

4.2. Number of classes

The next step was to determine the number of classes that were required to adequately describe the data. Numerous trials were performed on the acoustic impedance training set with the time window 1080-1100 ms, varying the numbers of classes to be determined by the network. Figures 10.12-10.14 show the comparison of three maps. The number of classes in each map is 10, 7 and 14. The weight vector for each class is given at the bottom of each figure. The waveforms show similar characteristics but the order of the traces is a little different. The major channel has been predicted by classes 1 and 2 for each case. The surrounding high porosity area of well 09-17 and 13-16 is also assigned to classes 1 and 2. However, the order of the classes is a little different. In Figure 10.13, wells 11-08 and 01-17 are predicted by class 3, however, they are in class 14 in the fourteen class map (Figure 10.14) and in class 5 in the ten class map (Figure 10.12). The map showing fourteen classes appears to have provided a less noisy classification.

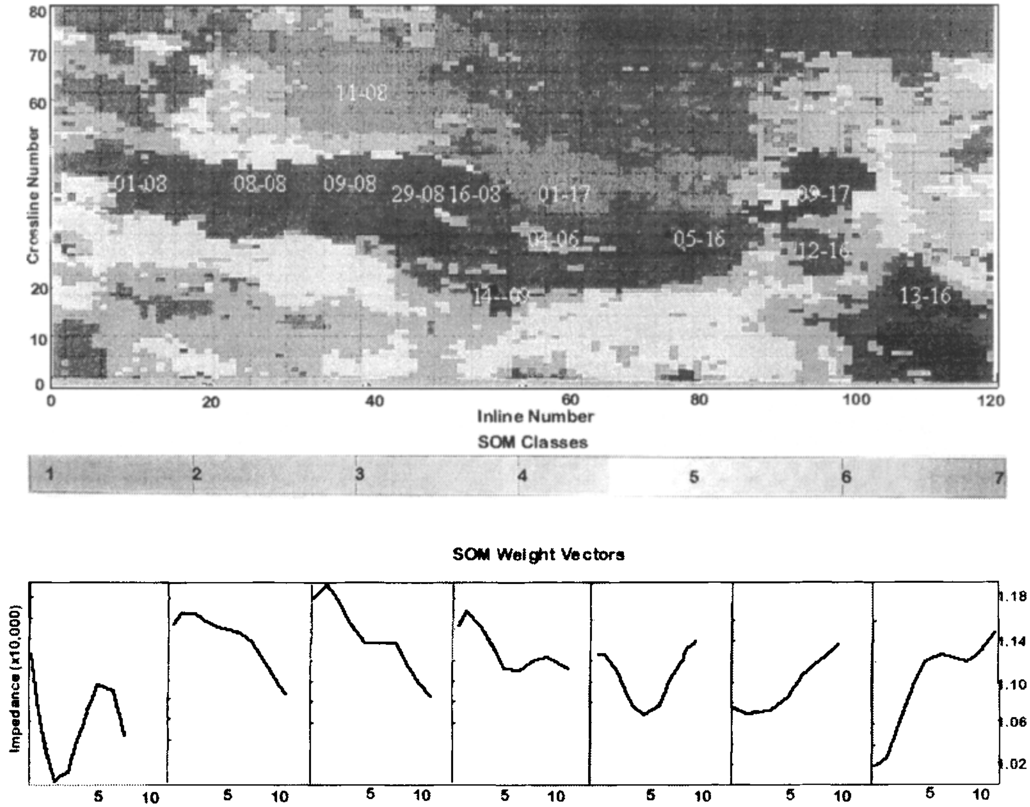


Figure 10.13. Results from using 7 classes for the acoustic impedance data between 1080 and 1100 ms.

5. CONCLUSIONS

The unsupervised SOM classification and horizon tracking were successful for the data set we tested. The self-organizing map network captured the features and patterns that were embedded in the seismic response. The horizon-tracking algorithm provided good resolution and continuity for interpreters. The horizon-tracking algorithms were very stable and could be applied as a pre-processing step to improve the seismic interpretations.

The unsupervised classification can also capture information contained in portions of the seismic traces. The results show the SOM algorithm can create a consistent classification at different time windows. In our data set, acoustic impedance produced classifications that correlated more closely with known geological properties like porosity.

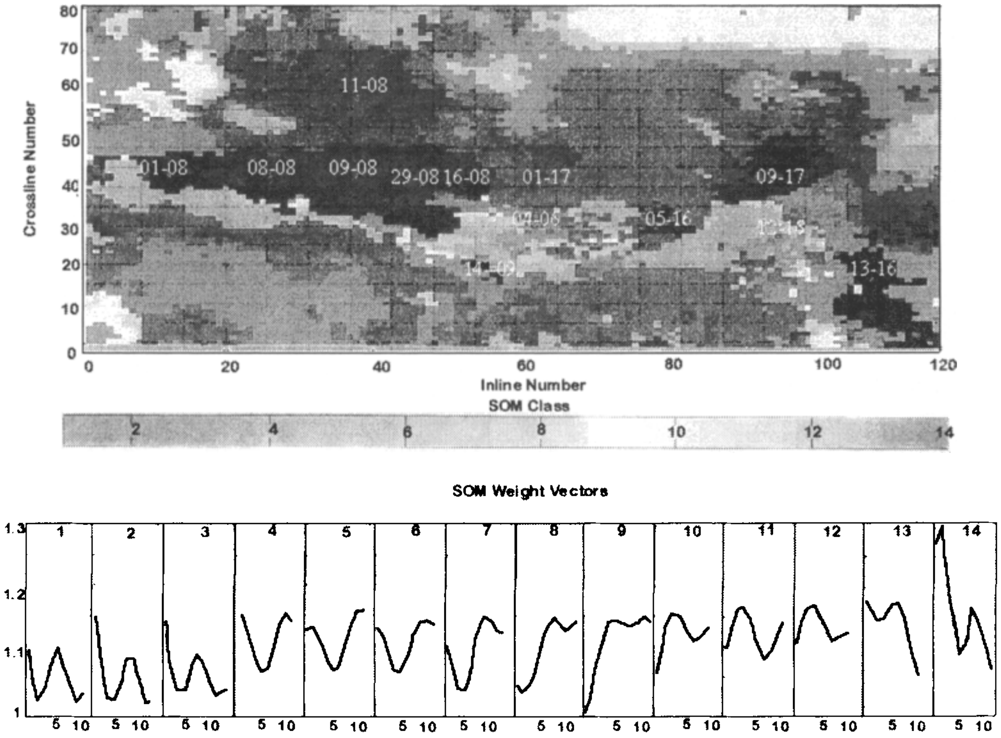


Figure 10.14. Results from using 14 classes for the acoustic impedance data between 1080 and 1100 ms.

REFERENCES

Hampson, D., Schuelke, J., and Quieren, J., 1999, Use of multi-attribute transforms to predict log properties from seismic data: submitted to Geophysics.

Harrigan. E, and Durrani. T.S., 1991, Automated horizon picking by multiple target tracking: 53^{rd} EAEG proceedings, 440-441.

Huang, K., 1990, Self-organizing neural network for picking seismic horizons: 60th Soc. Exploration Geophysicists, 313-316.

Chapter 11
Permeability Estimation with an RBF Network and Levenberg-Marquardt Learning

Fred K. Boadu

1. INTRODUCTION

Computational neural networks are essentially highly connected arrays of elementary processors called neurons and are particularly useful for solving complex decision problems that may not be well understood physically. Such problems abound in applied geophysics. There are an increasing number of scientific and engineering problems for which no direct algorithmic solution exists, but for which desired responses of problem examples are available. Neural networks can be used to solve such problems and offer the possibility of finding input-output correlations of essentially arbitrary complexity that have geological significance. Unlike traditional regression models incorporating a fixed algorithm to solve a particular problem, neural networks utilize a learning technique to develop a desirable solution such that the network is flexible and adaptive to different data sets. The associative relationship between input attributes and the specific output parameters is optimized without the constraint of *a priori* information. The use of neural networks to approximate non-linear relationships is particularly appealing in the context of geophysical data analysis. Neural networks have been criticized because they appear to operate like black boxes. However, the enormous quantity of knowledge or information they encode is available for more investigative work. Neural networks have the following useful features: (1) they have excellent generalized mapping capabilities; (2) they respond with high speed of input signals; (3) they filter noise from data; and (4) they can perform classification as well as function approximation. However, these methods have some drawbacks: (1) training is computationally intensive; (2) they are data intensive; (3) they have a tendency to be over-trained. Their usefulness, however, outweighs their drawbacks as the problems associated with them can be minimized by following careful procedures.

The usefulness of neural networks as a versatile tool in solving some otherwise difficult geophysical problems is demonstrated in this chapter. The networks are used to describe a reliable functional relationship between an easily measured geophysical parameter (velocity) and an elusive petrophysical parameter (permeability). The most influential parameters affecting permeability are also assessed making critical use of the neural network-based performance - prediction capability.

2. RELATIONSHIP BETWEEN SEISMIC AND PETROPHYSICAL PARAMETERS

One useful source of information for analysis and interpretation of geophysical data is the functional relationship between measured seismic parameters (velocity and attenuation) and petrophysical properties (porosity, permeability, grain size etc.). Seismic parameters can be measured or computed independently for a given fluid saturated elastic or poroelastic medium with known petrophysical properties using existing petrophysical models. However, neither the functional relationship between the parameters and the properties nor the exact underlying physical mechanism leading to the relationship is known with significant reliability. At best, we can conceive it to be highly non-linear. Rather than trying to derive functional relationships using some forms of approximations, a scheme to estimate the functional forms from the measured or computed data themselves will be developed. Use of a neural network is the recourse, and it can help determine highly non-linear functions that best fit those relationships. For estimation of linear relations, a neural network is overkill.

Neural networks are useful when searching for particular unknown nonlinear relationships or to approximate certain complicated data-generating mechanisms. Such relationships will be estimated (or learned) from a set of observations (geophysical parameters), computed or measured, and petrophysical data both of which constitute the learning set. This learning can be done on-line, allowing for on-line adaptation of the model as new data for the system become available. When the relationship is learned, it will remain constant and can then be used to predict or estimate values of petrophysical properties when only seismic information is available. The ability of some neural networks to approximate non-linear relationships justifies their use in characterizing complex systems such as fluid saturated poroelastic media. The choice of neural network architecture for such highly non-linear problems is the radial basis functions (RBF) network (Chen et al., 1991). The RBF networks are radial-distribution functions that have centers and widths expressed in terms of n-dimensional space defined by the input data vectors. These Gaussian basis functions produce a non-zero response only when an input vector falls within a small localized region of this space centered on the mean and within the specified width of the basis function. In general, the underlying learning algorithms of the RBF networks are fast, and since they are based on linear algebra, they are guaranteed to find the global optima (Chen et al., 1991). RBF networks are described in Chapters 5 and 16 in more detail.

Figure 11.1 illustrates a schematic representation of the RBF network with n inputs and a scalar output. Implementation of a mapping f_r: $\mathbf{R}^n \rightarrow \mathbf{R}$ by the network is described by the following equation (Chen et al., 1991),

$$f_r(x) = \lambda_0 + \sum_{i=1}^{n_r} \lambda_i \Phi(\|x - c_i\|), \tag{11.1}$$

where $x \in \mathbf{R}^n$ is the input vector, $\varphi(.)$ is a given function from $\mathbf{R}^+$ to $\mathbf{R}$, $\|.\|$ denotes the Euclidean norm, λ_i, $0 \leq I \leq n_r$, are the weights of the output node, c_i $(0 \leq I \leq n_r)$ are known as the RBF centers and n_r is the number of centers.

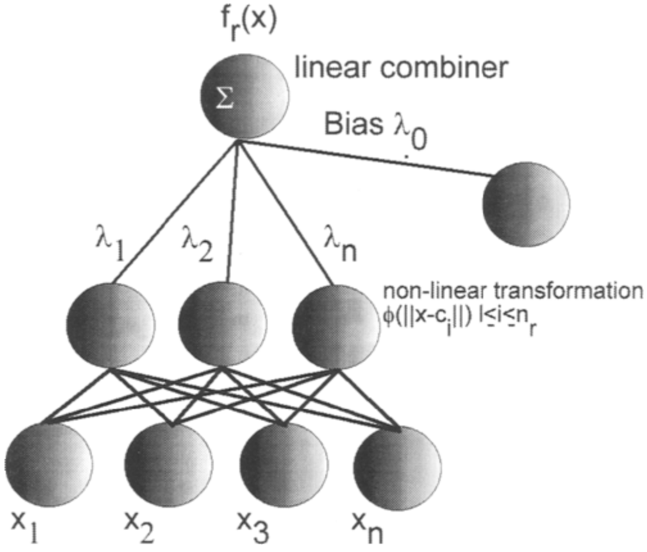


Figure 11.1. A schematic illustration of a RBF network with *n* inputs and one linear output.

In the RBF network, the functional form φ(.) and the centers c_i are assumed to be fixed. Given a set of input *x(t)*, and the corresponding output *d(t)* for t=1 to *n*, the values of the weights λ_i can be determined using the least-squares method. Careful consideration should be given to the choice of the functional form and the centers. One of the common functions used is the Gaussian function

$$\Phi(\|x - c_i\|) = \exp\left(-\frac{\|x - c_i\|^2}{\sigma^2_i} \right), \qquad (11.2)$$

where σ_i is a constant which determines the width of the input space of the *i*-th node. In practice the centers are normally chosen from the input data (Chen et al., 1991). The target function can be learned by specifying the n_r, σ_i, **x** and λ_i parameters. The localized nature of the fit makes the solution generated by the network clearly comprehensive and easy to relate the behavior of the fit to the learning set.

2.1. RBF network training

The objective here is to fit a non-linear function to a set of data comprised of seismic and petrophysical information. We presume that the fundamental physical mechanism underlying the relation between seismic parameters and the petrophysical properties is based on rock physics phenomenon. The RBF method is fully data-adaptive and as such all the network parameters must in some way be specified from the learning data set. The construction and optimization of the network involves several stages: First, an arbitrary number of Gaussian basis functions have to be selected; the selection procedure is based on the number of n-dimensional sample points in the learning set. Typically, an n-dimensional K-means clustering is used to exploit the natural clustering of the input data to locate the means (i.e., the centers) of the selected number of nodes such that their average Euclidean distances from all the input data are minimized. There should be fewer nodes than the sample points for the system to be sufficiently over-determined to obtain a smooth curve through the points. The output from this arbitrary chosen number of radial basis functions are then linearly correlated via least squares to the supplied target or output data to obtain the connection weights. The final stage in the training and optimization of the network involves systematically varying the number of clusters and the overlap parameters to achieve optimum fit to the training or learning data set.

In all the examples illustrated here, the network was trained using a subset of the full data database and their predictive capability was assessed using a completely independent test data set. This test set contains data that had not been exposed to the network during training and the data are selected randomly from the database used in the analysis.

2.2. Predicting hydraulic properties from seismic information: Relation between velocity and permeability

One of the most elusive problems in geophysics is establishing relations between measurable geophysical properties, for example, velocity or attenuation and petrophysical parameters, such as porosity and permeability. Due to refined improvements in the resolution and accuracy of seismic images of reservoirs as a result of advanced acquisition and processing techniques, there is heightened interest in the inversion of seismic data into petrophysical properties. For example, Han et al. (1986) found a good correlation between seismic compressional-wave velocity (V_p), porosity (φ) and clay content (V_{cl}) via the relation,

$$V_p = 5.59 - 6.39\Phi - 2.18V_{cl}. \tag{11.3}$$

Given the relation that exists between porosity and permeability, coupled with the strong relation between velocity, porosity and clay content, one would intuitively expect a relation between velocity and permeability. The factors that would cause velocity decrease, that is, increase in porosity and clay content, have the opposite effects on permeability. The importance of using geophysical parameters to predict permeability and the lack of existing relationship warrants further investigations and studies. An attempt is made in this chapter to establish a predictive relation between compressional velocity and permeability using the function approximation capability of computational neural networks. A number of researchers have attempted to relate seismic attenuation of rock to their permeability (e.g., Klimentos and McCann, 1990; Boadu, 1997). Establishing a relation between velocity and permeability, however, is a formidable task although such a functional relationship is needed to convert easily measured velocity values to permeability values. This can be useful in estimating the distribution of permeability away from wells when suitable corrections are applied. Klimentos and McCann (1990) have provided detailed measurements of velocity, attenuation and other petrophysical properties including permeability and porosity of rocks. Though the relationship is non-linear, we expect this to be a manifestation of the complex underlying rock physics phenomenon. The contention here is that if permeability is related to clay content and porosity and these same parameters are related to velocity (Han et al., 1986), then there must be a relationship between permeability and velocity. This relation has been given little attention and not explored. In principle we can develop theoretical relations using petrophysics as a guide (Boadu, 2000). Here, we will estimate this non-linear relation from available measured data. Neural networks, in particular RBF, which is useful in function approximation, can be very helpful in the determination of non-linear functions that best fit the relationship.

We first establish a relationship using least-squares (L-S) polynomial fitting between permeability and velocity employing 27 samples (Klimentos and McCann, 1990). The RBF is then used to establish the relation from the same data set. The L-S fit is a polynomial of fourth order and is described by the equation:

$$K = 95 - 1457V_p + 8200V^2{}_p - 20793V^3{}_p + 19385V^4{}_p, \tag{11.4}$$

where the permeability K is in millidarcies and V_p is the compressional wave velocity in km/sec. The comparison of the least-squares (L-S) polynomial fit with the RBF function approximation is illustrated in figure 11.2. The RBF fit gives a compelling coefficient of determination value (R^2) of 0.83. This implies that about 83% of the variance in the prediction of the permeability is explained by the RBF model. The L-S fit, however gives an R^2 value of 0.6, indicating that the model explains only 60% of the variations in the prediction. The RBF has learned a non-linear functional relationship between a seismic parameter (velocity) and a petrophysical parameter (permeability) using laboratory measurements. In comparing the two predictions, the output of the network is obviously statistically significant and, therefore, the model of choice. This illustrates one of the capabilities and usefulness of a neural network.

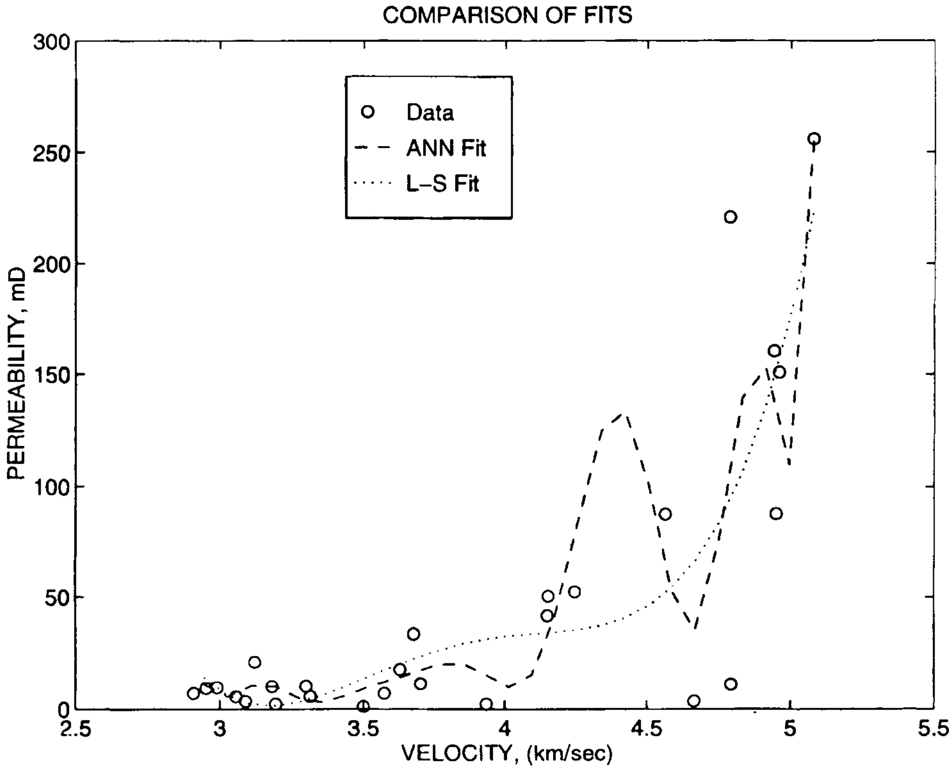


Figure 11.2. Comparison of the outputs of least-squares polynomial fit and RBF (ANN Fit) network function approximation.

3. PARAMETERS THAT AFFECT PERMEABILITY : POROSITY, GRAIN SIZE, CLAY CONTENT

One of the most important petrophysical parameters for characterizing a productive reservoir is permeability. Current practice in the oil industry requires a fluid flow test to yield reliable permeability values. Such tests are prohibitively expensive and a less expensive alternative is to estimate permeability from textural and compositional properties such as grain size, porosity and clay content. Using core data from Klimentos and McCann (1990), an attempt is made here to relate porosity, grain size and clay content to permeability through a regression model and compare its output with the output of the neural network. These relationships can then be applied to well-log data at a much larger scale. The effects of the indicators, porosity, grain size and clay content, are highly coupled and bear a non-linear relationship with permeability.

A commonly accepted relationship between grain size and permeability (hydraulic conductivity) was proposed by (Hazen, 1911) and given as:

$$K = Ad_{10}^{2}, \tag{11.5}$$

where K is the hydraulic conductivity in cm/s, A is a constant and d_{10} is the effective diameter defined as the value in the grain size distribution curve where 10% of the grains are finer. To account for the distribution of the grain size curve, Masch and Denny (1966) used the median grain size (d_{50}) as the distribution's representative size in an endeavor to correlate permeability with grain size. Krumbein and Monk (1942) expressed the permeability k of unconsolidated sands with lognormal grain size distribution functions with approximately 40% porosity by an empirical equation of the form:

$$k = 760 d_{w}^{2} \exp^{-1.31\sigma_{\psi}}, \tag{11.6}$$

where d_W is the geometric mean diameter (by weight) in millimeters and σ_ψ is the standard deviation of the ψ distribution function ($\psi = -\log_2 d$, for d in millimeters). The introduction of ψ converts the lognormal distribution function for the grain diameters into a normal distribution function in ψ. Berg (1970) modified the equation of Krumbein and Monk (1942) to account for the variation in porosity and determined the permeability variation with porosity of different systematic packing of uniform spheres using a semi-theoretical/empirical method.

The hydraulically based Kozeny-Carman (K-C) model has received great attention and relates the permeability to the porosity and grain size (e.g., Bear, 1972):

$$K = \left(\frac{\rho_w g}{\mu} \right) \frac{\Phi^3}{(1-\Phi)^2} \left(\frac{d_m^{2}}{180} \right), \tag{11.7}$$

where K is the hydraulic conductivity, ρ_w is the fluid density, μ is the fluid viscosity, φ is the porosity and d_m is a representative grain size. The choice of the representative grain size is critical to the successful prediction of hydraulic conductivity from the grain size distribution. In applying this equation, a fixed value of d_m is typically chosen to represent the entire range of grain sizes. Koltermann and Gorelick (1995) assert that the use of the geometric mean overpredicts hydraulic conductivity by several orders of magnitude for soils with significant fines content. In contrast, the authors indicate that the harmonic mean grain size representation underpredicts K by several orders of magnitude for soils with lesser fines content. Their reasoning is that, overall, the harmonic mean puts greater weight on smaller grain sizes while the geometric mean puts heavier weight on larger sizes. The percentage of clay plays a significant role. In some rocks and soils, clay content exceeding 8% lowers the hydraulic conductivity as the clay particles fill the voids between the sand particles and control the hydraulic behavior of the soil or rock. The efforts by various researchers described above indicate that permeability is affected by these three petrophysical factors: porosity, grain-size and clay content. It is crucial, therefore, to establish how these factors influence the permeability and the extent of their influence. Some valid questions whose answers would be useful are: Can we reliably predict permeability from knowledge of the petrophysical factors that can easily be obtained from cores and well logs? What is the most influential petrophysical factor that affects permeability? In the next section we attempt to answer these fundamental questions by exploiting the generalization and predictive capabilities of a neural network.

4. NEURAL NETWORK MODELING OF PERMEABILITY DATA

The database used in this work is based on the experimental data provided by Klimentos and McCann (1990). A fully connected, three layer, feed-forward neural network was used in this study as shown in Figure 11.3.

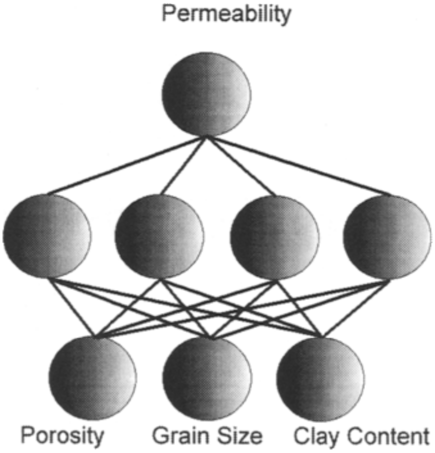


Figure 11.3. A schematic illustration of feed-forward (MLP) neural network.

The multivariate statistical methods used by several researchers to establish a relationship between some of their parameters characterizing the rock properties and seismic parameters are often complex and require the important parameters to be known for its formulation. On the other hand, the modeling process of neural networks is more direct and capable of capturing complex non-linear interactions between input and output in a physical system. During training, irrelevant input variables are assigned low connection weights that may be discarded from the data. In this study neural networks are trained on measured laboratory data and are trained to deal with and handle inherently noisy, inaccurate and insufficient data. The Levenberg-Marquardt (LM) training algorithm is utilized as it has been found to be more efficient and reasonably more accurate than the traditional gradient descent back-propagation (Hassoun, 1995).

The LM optimization method provides an alternative and more efficient way of minimizing the sum-square-error E. The back-propagation algorithm is based on the gradient descent technique that has a major drawback of requiring a large number of steps before converging to a solution. A reasonable increase in the convergence rate has been noted by Hassoun (1995) when the quasi-Newton optimization algorithm is used. On the other hand, an important limitation of the quasi-

Newton method is that it requires a good initial guess for convergence. The suggested alternative, the LM routine (see Chapter 5), is essentially an interpolation between the quasi-Newton and gradient descent methods and successfully hybridizes the useful properties of the two methods for optimal performance. The inherent difficulty in selecting the appropriate momentum and learning rate terms in the conventional back-propagation algorithm is overcome in this scheme.

Consider the sum-of-squares error function as above in the form

$$\bar{E} = \frac{1}{2}\sum_{m}(\varepsilon^{m})^{2} = \frac{1}{2}\|\bar{\varepsilon}\|^{2}, \tag{11.8}$$

where ε^m represents the error associated with the *m*th input pattern, and ε is a vector with elements ε^m. For small perturbations in the weights **w**, the error vector can be expanded to a first order via the Taylor series expansion:

$$\varepsilon(\vec{w}_{new}) = \varepsilon(\vec{w}_{old}) + G(\vec{w}_{new} - \vec{w}_{old}), \tag{11.9}$$

where w_{old} and w_{new} indicate current and old points in weight space, respectively, and elements in the matrix **G** are defined as

$$\mathbf{G}_{mi} = \frac{\partial \varepsilon^{m}}{\partial \vec{w}_{i}}, \tag{11.10}$$

Thus the error function defined above can be written as

$$E = \frac{1}{2}\|\varepsilon(\vec{w}_{old}) + G(\vec{w}_{new} - \vec{w}_{old})\|^{2}. \tag{11.11}$$

If this error function is minimized with respect to the new weights $\mathbf{w}_{new}$ we obtain

$$\vec{w}_{new} = \vec{w}_{old} - (G^{t}G)^{-1}G^{t}\varepsilon(\vec{w}_{old}). \tag{11.12}$$

The above formula can, in principle, be applied iteratively in an attempt to minimize the error function. Such an approach inherently poses a problem in that the step size (change in weights) could be large in which case the basic assumption (small change in weights) on which equation 11.11 was developed would no longer be valid. The LM algorithm addresses this problem by seeking to minimize the error function while simultaneously trying to keep the step size small enough to ensure that the linear approximation remains valid. To achieve this aim, the error function is modified into the form:

$$E_{\text{mod}} = \frac{1}{2}\|\varepsilon(\vec{w}_{old}) + G(\vec{w}_{new} - \vec{w}_{old})\|^{2} + \lambda\|\vec{w}_{new} - \vec{w}_{old}\|^{2}, \tag{11.13}$$

where the parameter λ governs the step size. Minimization of the modified error with respect to $\mathbf{w}_{\text{new}}$ gives

$$\vec{w}_{new} = \vec{w}_{old} - (G^T g + \lambda I)^{-1} G^T \varepsilon(\vec{w}_{old}), \qquad (11.14)$$

where I is the unit matrix.

The weight correction term in the LM scheme is obtained as

$$\Delta w_{jt} = (G^T G + \lambda I)^{-1} G^T e, \qquad (11.15)$$

where $\boldsymbol{e}$ is an error vector (difference between output and target values). For large values of the damping factor, equation 11.15 approximates the gradient descent of back-propagation while for small values it leads to the quasi-Newton method. The damping parameter is adjusted according to the nature of changes in the error as learning progresses. As the error gets smaller, λ is made bigger. If the error increases however, λ is made smaller. The choice of the damping factor is crucial to the convergence rate and stability of the learning process. In this work, the damping factor is chosen as 1.0 percent of the largest singular value in each iteration, which provided satisfactory results. In the training process of the network, it took the conventional back-propagation algorithm 4 minutes to train the network while the LM algorithm took only 47 seconds. The RMS errors were 0.187 for the LM and 0.371 for the conventional back-propagation. When using any iterative training procedure, a criterion must be available to decide when to stop the iterations. In this work training continued until the sum-squared error reached an acceptable value (0.01) for the entire training set or after a fixed number (1000) of training cycles had been reached.

As noted by Hassoun (1995), proper selection of a training set with the right type of data preprocessing and an appropriate number of data points as input to a neural network may outrank the importance of the network design parameters. The input attributes are: (1) porosity, (2) mean grain size and (3) clay content. The output or target parameter is the measured permeability. In designing the network a good set of data to be used for training the net was processed and used as input to the network. The data were divided into two sets. For the first set, about 80 percent of the whole dataset selected randomly constitute the set used to train the network. The remaining 20 percent were used test the net. The input values to be supplied to the net were pre-processed by a suitable transformation to lie in the range 0-1. The normalization techniques used to transform or normalize the input training data x_k (a vector composed of just the kth feature in the training set) in the interval [β,α] is expressed as:

$$y_k = (\alpha - \beta) \cdot \frac{y_k - y_{\min}}{y_{\max} - y_{\min}} + \beta, \qquad (11.16)$$

where y_{min} and y_{max} are the respective values of the minimum and maximum elements in the training and the testing data.

For a finite number of examples, the simplest network, the one with the fewest number of weights, that satisfies all input-output relations given by the training set, might be expected to have the best generalization properties (Dowla and Rogers, 1995). The number of examples-to-weight ratio (EWR) in the training process was restricted to values greater than 9, which is close to the value of 10 recommended by Dowla and Rogers (1995). All computations were performed on the SPARC-20 Unix workstation using the MATLAB® programming language.

4.1. Data Analysis and Interpretation

In this section, we illustrate the predictive capability of the neural network and compare it with the least-squares prediction. The useful exploitation of the stored information in the network and its use in solving problems related to rock physics is also described. The relative importance of porosity, grain-size, and clay content as petrophysical parameters in influencing permeability is addressed and analyzed. The regression equations are developed to predict the permeability values using porosity, grain-size and clay content as descriptors. The regression model relates the permeability values to the descriptors via the following equation (Draper and Smith, 1981):

$$P = \alpha_0 + \alpha_1 X_1 + \alpha_2 X_2 + \ldots + \alpha_n X_n, \tag{11.17}$$

where P is the computed permeability value, α_n is the coefficient determined by the regression analysis, and X_n is the value of the descriptor or petrophysical parameter. The three descriptors were used to develop a regression model for comparison with that of neural network. The resulting regression equation relating permeability K (mD) to the petrophysical parameters, porosity, grain-size and clay content is given as:

$$K = -63 + 6.15\Phi + 0.42D - 7.15C, \tag{11.18}$$

with a correlation coefficient (R^2) of 0.59, where φ is the porosity (%), D is the mean grain size (μm), and C is the clay content (%). The regression equation (11.18) is then used to predict permeability values given the input descriptors or petrophysical parameters. In the regression model, all the available data were used to obtain the following equation. For the neural network modeling, however, part of the data (~80%) was used for training of the network and the remaining used in assessing its external prediction potential.

A plot of the measured versus predicted permeability values using the L-S and the neural network models is shown in Figure 11.4.

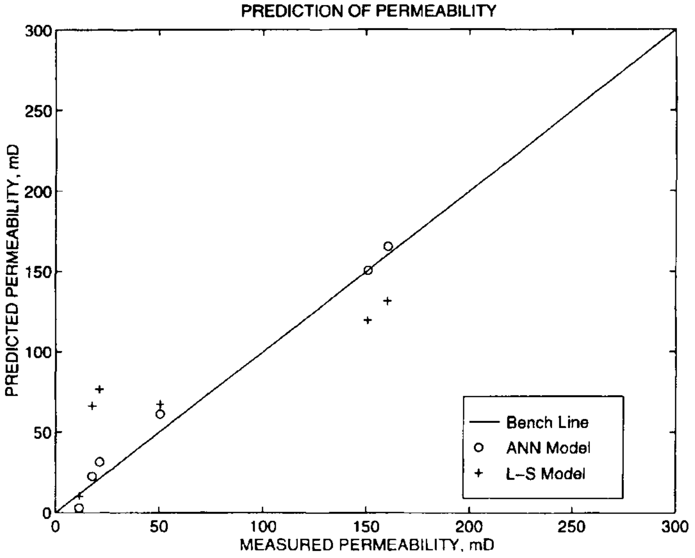


Figure 11.4. Comparison of L-S and neural network (ANN Model) predictions of permeability.

The bench line is the decision line along which the measured and predicted attenuation values are equal. Points falling along or close to this line indicate accurate reasonable predictions. The neural network model predictions matched the permeability values reasonably well with a correlation coefficient (R^2) of 0.94 and standard error of estimate of 16.8. This is deemed a very good match considering that the neural network has not been exposed to these data values. On the other hand, the L-S model has been exposed to the testing data (as the testing data were part of the data used in regression modeling), but provides a weaker prediction capability in comparison to the neural network model. The standard error of estimate for the least squares (L-S) model is 60.5. Each petrophysical parameter has a level of significant contribution to the overall permeability. However, the degree of influence of each parameter may not be obvious. A method is provided to make an inference on the relative importance or the degree of influence of each parameter on the permeability values via the neural networks. This involves analysis of the weights of the fully trained network and the procedure is described below.

4.2. Assessing the relative importance of individual input attributes

The relative importance of the individual input attributes as to their influence on permeability was evaluated using the scheme developed by Garson (1991). Though the scheme is approximate, it nevertheless provides intelligible and intuitive insights regarding the internal processing and operation of neural networks. It has been used successfully to decipher the relevant relationships between the physical parameters characterizing rock properties. The method basically involves partitioning the hidden-output connection weights of each hidden node into components associated with each input node. The weights along the paths linking the input to the output node contain relevant information regarding the relative predictive importance of the input attributes: the weights can be used to partition the sum effects of the output layer. The connection weights of the neural network after training are shown in table 11.2.

Table 11.2
Optimal connection weights

Hidden nodes	Weights			
(1)	Input #1 (2)	Input #2 (3)	Input #3 (4)	Output (5)
1	23.040	-3.670	11.102	9.259
2	-19.218	-1.655	13.841	-3.356
3	-18.658	-2.340	11.890	-5.224
4	-9.413	-3.163	2.685	-9.408

The algorithm for estimating the relative importance is as follows:

1. For each node *i* in the hidden layer, form the products of the absolute value of the hidden-output layer connection weight and the absolute value of the input-hidden layer connection weight. Perform the operation for each input variable *j*. The resulting products Γ_{ij} are presented in Table 11.3.
2. For each hidden node, divide the product Γ_{ij} by the sum of such quantities for all the input variables to obtain Φ_{ij}. As an example for the first hidden node, $\Phi_{11} = \Gamma_{11}/(\Gamma_{11} + \Gamma_{12} + \Gamma_{13}) = 0.1303$.
3. The quantities Φ_{ij} obtained from the previous computations are summed to form X_j. Thus we have for example, $x_1 = \Phi_{11} + \Phi_{21} + \Phi_{31} + \Phi_{41}$. The results are shown in Table 11.4.

Table 11.3
Elements of matrix of products Γ_{ij}

Hidden PE (1)	Input #1 (2)	Input #2 (3)	Input #3 (4)
1	21.333	3.399	10.280
2	64.509	5.557	46.462
3	9.874	1.238	6.293
4	8.856	2.976	25.269

Table 11.4
Elements of Φ_{ij} and x_j

Hidden PEs (1)	Input #1 (2)	Input #2 (3)	Input #3 (4)
1	$\Phi_{11} = 0.6093$	$\Phi_{12} = 0.0970$	$\Phi_{13} = 0.2936$
2	$\Phi_{21} = 0.5535$	$\Phi_{22} = 0.0477$	$\Phi_{23} = 0.3987$
3	$\Phi_{31} = 0.5673$	$\Phi_{32} = 0.0712$	$\Phi_{33} = 0.3615$
4	$\Phi_{41} = 0.2387$	$\Phi_{42} = 0.0802$	$\Phi_{43} = 0.6810$
{Sum}	$x_1 = 1.9688$	$x_2 = 0.2961$	$x_3 = 1.7348$

4. Divide x_j by the sum for all input variables. The result in terms of percentages gives the relative importance or influence of all output weights attributable to the given input variable. For example for the first input node, the relative importance is equal to $(x_1/(x_1 + x_2 + x_3 + x_4)) \times 100 = 32.43\%$. For the four input nodes, the results are given in Table 11.5. It should be noted that the biases are not factored into the partitioning process, as they do not affect the outcome of the process (Garson, 1991).

Table 11.5
Relative importance of input petrophysical parameters

Result (1)	Input # 1 (porosity)	Input # 2 (grain size)	Input # 3 (clay content)
Relative importance	49.22%	7.40%	43.33%

These numerical results indicate that the most influential petrophysical parameter affecting permeability is porosity (49%), followed closely by clay content (43%). The average grain-size as a parameter has a minimal influence on the permeability. This fact fortifies the argument of Boadu (2000) that for grain-size to be valuable in permeability prediction, a parameter, such as fractal dimension describing the distribution, is more useful than say the average or median grain size. The porosity and clay content should be given similar importance in permeability analysis using petrophysical data.

5. SUMMARY AND CONCLUSIONS

Laboratory data was utilized to establish relations between seismic and petrophysical parameter using a radial basis function architecture as a versatile function approximator. The network-derived relation was used predict permeability from velocity measurements with reasonable accuracy. The predictive capability of the network was also exploited to predict permeability values from available petrophysical parameters, porosity, grain-size and clay content. Using valuable information stored in the fully trained network weights, the relative weight of influence of each parameter on permeability was assessed. The results indicate that the most influential petrophysical parameter affecting permeability is porosity (49%), followed closely by clay content (43%), with the average grain-size having a minimal influence. These pieces of information are useful in predicting an elusive subsurface property such as permeability from geophysical and petrophysical information.

Computational neural networks are useful in providing new versatile tools in solving geophysical problem. If one subscribes to the notion that the approximate solution to the right problem is more useful than the exact solution to the wrong problem, then neural networks are not just passing paradigms but versatile tools in solving geophysical problems.

REFERENCES

Bear, J., 1972, Dynamics of Fluid in Porous Media: Elsevier, New York.

Berg, R., 1970, Method for determining permeability from reservoir rock properties: Trans. Gulf Coast Assoc. Geol. Soc., **20**: 303-317.

Boadu, F., 1997, Rock properties and seismic attenuation: Neural network analysis: Pure and Applied Geophysics, **149**, 507-524.

Boadu, F., 2000, Hydraulic conductivity of soils from grain-size distribution: New models: J. of Geotechnical and Geoenvironmental Engineering, 126, 739-745.

Chen, S., Cowan, C. and Grant, P., 1991, Orthogonal least-Squares learning algorithm for radial basis function networks: IEEE Trans. on Neural Networks, **2**, 302-309.

Dowla, F.W., and Rogers, L.L, 1995, Solving Problems in Environmental Engineering and Geosciences with Computational Neural Networks: MIT Press.

Draper, N., and Smith, H., 1981, Applied Regression Analysis: John Wiley and Sons Inc.

Garrett, J., 1994, Where and why computational neural networks are applicable in civil engineering: J. Geotechnical Eng. **8**, 129-130.

Garson, G., 1991, Interpreting neural-network connection weights: AI Expert, 7, 47-51.

Han, D., Nur, A., and Morgan, D., 1986, Effects of porosity and clay content on wave velocities in sandstone: Geophysics, **51**, 2093-2017.

Hassoun, M., 1995, Fundamentals of Artificial Neural Networks: MIT Press.

Hazen, A., 1911, Discussion of "Dams on sand foundations" by A.C. Koenig: Trans. Am. Soc. Civ. Eng., **73**,199.

Klimentos, T., and McCann, C., 1990, Relationships among compressional wave attenuation, porosity, clay content and permeability in sandstones: Geophysics, **55**, 998-1014.

Koltermann, C. and Gorelick, S., 1995: Fractional packing model for hydraulic conductivity derived from sediment mixtures: Water Resources Res., **31**, 3283-3297.

Krumbein, W., and Monk, G., 1942, Permeability as a function of the size parameters of unconsolidated sand: Am. Inst. Min. Metall. Eng., Tech. Pub., 153-163.

Masch, F., and Denny, K., 1966, Grain-size distribution and its effects on the permeability of unconsolidated sand: Water Resources Research, **2**, 665-677.

Chapter 12
Caianiello Neural Network Method For Geophysical Inverse Problems

Li-Yun Fu

1. INTRODUCTION

Information-bearing geophysical signals have a great variety of varying physical realizations. The effects resulting from simple mathematical models may be collected into a catalog of master curves or overall trends for use in comparison with observed effects. Geophysical inverse problems are strongly related to inexact data (e.g., information-incomplete, information-overlapping, and noise-contaminated) and ambiguous physical relationships, which leads to nonuniqueness, instability, and uncertainty of the inversion. The ambiguous dependence of observed geophysical data related to subsurface physical properties suggests that geophysical inverse problems be characterized by both a deterministic mechanism and statistical behavior. The optimal inversion method is the one with the ability to aptly merge certain deterministic physical mechanisms into a statistical algorithm.

Caianiello neural networks are based on the Caianiello neuron equation, a convolutional neuron model. In contrast to conventional neural networks based on the McCulloch-Pitts model, a dot-product neuron, Caianiello networks have many advantages, for example, time-varying signal processing and block frequency-domain implementation with fast Fourier transforms (FFTs). Caianiello neural networks provide the ability to deterministically incorporate geophysically meaningful models into the statistical networks for inversion. The Caianiello neural network method for geophysical inverse problems consists of neural wavelet estimation, input signal reconstruction, and nonlinear factor optimization. These algorithms result in the inverse-operator-based inversion and forward-operator-based reconstruction for solving a wide range of geophysical inverse problems.

A geophysical system is generally composed of both the minimal set of physical parameters describing the system and the physical relationships relating the parameters to the results of the measurements of the system. Generalized geophysical inversion can be viewed as an inference about the parameters based on the physical relationships from observed geophysical data. On the other hand, some parameter properties of the system are statistically described. This results in a probabilistic description of the measured results (Tarantola and Valette, 1982). Therefore, geophysical inversion is a physical problem that is characterized by both a deterministic mechanism and statistical behavior. From the viewpoint of information theory, most geophysical processes are irreversible, in which information leakage cannot be reconstructed reliably. Only a limited amount of information is available that is contaminated with noise. Therefore, geophysical inverse problems almost always lack uniqueness, stability, and certainty. These problems are strongly related to inexact observed data (e.g., information-incomplete, information-overlapping, and noise-contaminated) and ambiguous physical relationships. Information recovery by inversion has to resort to integration of data from several sources. In summary, generalized geophysical inversion tends to establish a picture of the subsurface by using both deterministic and statistical approaches to integrate and analyze vast amounts of data generated from different sensors. Each type of data has different scales of resolution and different spatial distributions. A comprehensive integrated information system with adaptive fast history matching is needed to provide not only an extensive database to store information, but more importantly, a capacity with both deterministic and statistical methods for efficient and comprehensive data analysis in the problem-solving environment. A computational neural network system for time-varying signal processing, as an integrated approach proposed in this chapter, will provide significant potential for constructing this integrated systematic platform oriented to reservoir detecting, characterizing, and monitoring.

Most existing neural networks are based on the McCulloch-Pitts neuron model (McCulloch and Pitts, 1943). The input-output relation of the neuron is a dot-product operation that makes these networks unsuited to process the temporal information contained in the input signal. Network algorithms that use FFTs and corresponding block updating strategies for weights can efficiently solve the problem, but it is not easy to implement this scheme in the McCulloch-Pitts (MP) neural networks. In this study I construct a nontraditional neural network based on the Caianiello neuron equation (Caianiello, 1961), a convolutional model for a neuron's input-output relationship. This neural network has many advantages for solving problems in exploration geophysics. The most important for our applications is to process time signals adaptively and efficiently, to aptly merge some deterministic, geophysically meaningful models into the statistical network, and to implement the algorithm in the frequency domain using block-updating strategies with FFTs. The Caianiello neural network has been successfully incorporated with some deterministic petrophysical models for porosity and clay-content estimations (Fu, 1999a; Fu, 1999b) and with the Robinson seismic convolutional model (RSCM) for single-well-controlled seismic impedance inversion (Fu, 1995; Fu et al., 1997) and for multi-well-controlled seismic impedance inversion (Fu, 1997; Fu, 1998). In this chapter, the neural network will be extended to solve generalized geophysical inversions. The Caianiello neural network method for inverse problems consists of neural wavelet estimation, input signal reconstruction, and nonlinear factor optimization. These algorithms result in the inverse-operator-based inversion and forward-operator-based reconstruction for solving a wide range of geophysical inverse problems.

2. GENERALIZED GEOPHYSICAL INVERSION

2.1. Generalized geophysical model

A generalized geophysical system can be viewed as a real-time, topographic, and nonlinear mapping of the input space onto the output space by an internal representation of the earth system. The spatiotemporal integration of the system can be described as: at each point in space and time of the model, the signal is weighted by some coefficient and these values are added together. The kernel function specifying these weighting coefficients completely characterizes the earth system and could simply be the response of the earth system to a unit impulse $\delta(x,y,z,t)$. Strictly speaking, most geophysical processes are nonlinear (Lines and Treitel, 1984). The nonlinear transform f is always a formidable problem and has been studied extensively in a wide range of geophysical problems. For instance, some simple nonlinear transforms, featured with a nonconstant, bounded, monotonic, and continuous function, can be used to empirically model observed datasets. Consequently, a generalized geophysical model with a differentiable nonlinear function f can be depicted as

$$\phi(\mathbf{r},t)=f[\iint K(\mathbf{r},\mathbf{r}',t)s(\mathbf{r}',t)d\mathbf{r}'dt], \tag{12.1}$$

where $\mathbf{r}=(x,y,z)$, $\mathbf{r}'=(x',y',z')$, $s(\mathbf{r}',t)$ could be the input signal, objective function, or model parameters respectively, and correspondingly $\phi(\mathbf{r},t)$ is the output signal, imaging function, or model response. The kernel function $K(\mathbf{r},\mathbf{r}',t)$ is an information-detection and integration operator to $s(\mathbf{r}',t)$. For some nonlinear geophysical systems, the kernel function becomes $K(\mathbf{r},\mathbf{r}',t,s(\mathbf{r}',t))$.

For a geophysical system with spatiotemporal invariance, the generalized model of Eq. (12.1) can be simplified as a generalized convolutional equation

$$\phi(\mathbf{r},t)=f[K(\mathbf{r},t)*s(\mathbf{r},t)]=f[\iint s(\mathbf{r}',\tau)K(\mathbf{r}-\mathbf{r}',t-\tau)d\mathbf{r}'d\tau], \tag{12.2}$$

where $*$ is a spatiotemporal convolution symbol of operation. The model (12.2) makes a general framework to mathematically describe a geophysical system. For example, letting the transform $f(x)=x$ and taking the kernel function $K(\mathbf{r},\mathbf{r}',t)$ as the free space Green's function results in the following integral equation for wave propagation in the isotropic and elastic model of the earth.

$$\phi(\mathbf{r},t)=\iint s(\mathbf{r}',\tau)G(\mathbf{r}-\mathbf{r}',t-\tau)d\mathbf{r}'d\tau, \tag{12.3}$$

where $\phi(\mathbf{r},t)$ is a primary physical potential field related to either acoustic, electromagnetic or elastic waves, and $s(\mathbf{r}',t)$ is the source distribution ranged in a limited region. Eq. (12.3) has been widely used for inverse source problems. Similarly, one can take $K(\mathbf{r},\mathbf{r}',\omega)=(k^2(\mathbf{r}')-k_0^2)G(\mathbf{r},\mathbf{r}',\omega)$ to obtain the Lipmann-Schwinger equation with the Born approximation where k_0 is the background wavenumber. I will show later that Eq. (12.2) can be further extended for comprehensively analyzing various observed datasets to identify relationships and recognize patterns.

This physically meaningful kernel function $K(\mathbf{r},\mathbf{r}',t)$ contains the important geophysical parameters being investigated and other various effects of complex subsurface media. If the purpose of inversion is to extract the parameters, the simplification of the kernel function is indispensable for practical applications where we often make some compromises among medium complexity, method accuracy, and application possibility. For instance, the Robinson seismic convolutional model (Robinson, 1954, 1957; Robinson and Treitel, 1980) is an excellent example that has been widely used in exploration geophysics. In this chapter, I take the reduced geophysical model as an example to demonstrate a joint inversion scheme for model parameter estimation. The joint inversion strategy consists of the inverse-operator-based inversion and forward-operator-based reconstruction. It can be easily extended to solve a wide range of geophysical inverse problems with a physically meaningful forward model.

The geophysical inversion with Eq. (12.2), a Fredholm integral equation of the first kind, is a multidimensional deconvolution problem that involves the following two problems. First, the output signal $\phi(\mathbf{r},t)$ should be known over a sufficiently large region of the spatiotemporal frequency space, but the obtainable data from different measurements are always of finite spatiotemporal distributions. Information leakage in most geophysical processes is irreversible, which implies that the missing signals cannot be reconstructed from available data. For the inverse-operator-based inversion, the performance of inverse algorithms is impaired by noise taking over in the missing spatiotemporal frequencies. For the forward-operator-based inversion, due to the band-limited output $\phi(\mathbf{r},t)$, one can put any values in the missing spatiotemporal frequencies to produce infinity of different parameter models that, however, fit identical data. The solution of these problems, to some degree, resorts to multidisciplinary data integration. For instance, reservoir inversions incorporate seismic data (with surface, VSP, and interwell measurements), static well and core data, dynamic test data from production, and the geoscientist's knowledge about reservoirs. The second problem encountered with geophysical inversions is ill posedness and singularity.

2.2. Ill-posedness and singularity

Taking an operator expression of Eq. (12.1), we have

$$Ls(\mathbf{r},t)=\phi(\mathbf{r},t), \tag{12.4}$$

where L is an integral operator. The ill-posedness of an inverse problem with Eq. (12.4) can be defined that adding an arbitrarily small perturbation ε to $\phi(\mathbf{r},t)$ causes a considerably large error δ in $s(\mathbf{r},t)$, i.e. $L^{-1}[\phi(\mathbf{r},t)+\varepsilon]=s(\mathbf{r},t)+\delta$. The ill-posedness of inversion means $\delta >> \varepsilon$. Geophysical inverse problems are almost always ill-posed. This can be proved by the Riemann-Lebesgue theorem. Based on this theorem, if $K(x,y,\xi,\eta,\omega)$ is an integrable function with $a<\xi,\eta<b$, then

$$\lim_{\alpha\to\infty}\lim_{\beta\to\infty}\int_a^b\int_a^b K(x,y,\xi,\eta,\omega)\sin(\alpha\xi)\sin(\beta\eta)d\xi d\eta=0. \tag{12.5}$$

In this case, Eq. (12.1) with time invariance can be rewritten as

$$\begin{aligned}&\lim_{\alpha\to\infty}\lim_{\beta\to\infty}\int_a^b\int_a^b K(x,y,\xi,\eta,\omega)[s(\xi,\eta,\omega)+\sin(\alpha\xi)\sin(\beta\eta)]d\xi d\eta\\&\quad=\int_a^b\int_a^b K(x,y,\xi,\eta,\omega)s(\xi,\eta,\omega)d\xi d\eta=\phi(x,y,\omega).\end{aligned} \tag{12.6}$$

This result indicates that adding a sine component of infinite frequency to $s(\mathbf{r},t)$ leads to the same $\phi(\mathbf{r},t)$. In the case that α and β have finite values, for an arbitrary $\varepsilon > 0$, there exists a constant A for $\alpha, \beta \geq A$ such that

$$\int_a^b \int_a^b K(x,y,\xi,\eta,\omega)\sin(\alpha\xi)\sin(\beta\eta)d\xi d\eta < \varepsilon. \tag{12.7}$$

Thus

$$\begin{aligned}\int_a^b \int_a^b & K(x,y,\xi,\eta,\omega)[s(\xi,\eta,\omega)+\sin(\alpha\xi)\sin(\beta\eta)]d\xi d\eta \\ &= \int_a^b \int_a^b K(x,y,\xi,\eta,\omega)s(\xi,\eta,\omega)d\xi d\eta + \varepsilon_1 = \phi(x,y,\omega)+\varepsilon_1,\end{aligned} \tag{12.8}$$

where $|\varepsilon_1| < \varepsilon$. The above equation implies that perturbating $\phi(\mathbf{r},t)$ by a infinitely small value will increase $s(\mathbf{r},t)$ by a sine component with frequency of $\alpha, \beta \geq A$, a considerably large perturbation. This is the mathematical proof of the ill posedness of inversion. In the practical inversions, the infinitely small perturbation ε always exists in the observed data. The resulting errors will be increased in the inversion procedure and the inverse algorithm becomes unstable. Some stability conditions need to bound the magnitude of the objective function or to better condition the associated operator matrix. In this way, the inversion procedure is stable where a small change of the data will map to a physically acceptable error in the parameters to be inverted.

Singularity refers to the nonexistence of the inverse transform L^{-1} in Eq. (12.4). The singular problem of inversion depends on the kernel function properties. For instance, if $K(x,y,\xi,\eta,\omega)$ and $g(\xi,\eta,\omega)$ are orthogonal in $[a,b]$, i.e.

$$\int_a^b \int_a^b K(x,y,\xi,\eta,\omega)g(\xi,\eta,\omega)d\xi d\eta = 0, \tag{12.9}$$

we call $K(x,y,\xi,\eta,\omega)$ singular to $g(\xi,\eta,\omega)$. In this case, no information in $g(\xi,\eta,\omega)$ maps to the output $\phi(x,y,\omega)$. In general, the model $s(\xi,\eta,\omega)$ consists of the orthogonal component $g(\xi,\eta,\omega)$ and unorthogonal component $c(\xi,\eta,\omega)$. Thus,

$$\begin{aligned}\int_a^b \int_a^b & K(x,y,\xi,\eta,\omega)[g(\xi,\eta,\omega)+c(\xi,\eta,\omega)]d\xi d\eta \\ &= \int_a^b \int_a^b K(x,y,\xi,\eta,\omega)c(\xi,\eta,\omega)d\xi d\eta = \phi(x,y,\omega).\end{aligned} \tag{12.10}$$

Therefore, the orthogonal component cannot be recovered from the observed data $\phi(x,y,\omega)$. This may explain why the geophysical model parameters are often found to depend in an unstable way on the observed data. The solution of this problem, to some degree, resorts to multidisciplinary information integration.

Nonuniqueness, ill posedness, singularity, instability, and uncertainty are common inherently with geophysical inverse problems. In the sense of optimization, the least-squares procedure has been widely used to develop numerical algorithms for geophysical inversions. However, the above inherent problems prevent many of the classical inverse approaches from being used for the inversion of actual field recordings. Practical methods may be the joint application of deterministic and statistical approaches.

2.3. Statistical strategy

Due to these common problems related to most geophysical processes, such as inexact observed data, complex subsurface media, rock property variability, and ambiguous physical relationships, statistical techniques based mostly on Bayesian and kriging (including cokriging) methods have been extensively used for generalized geophysical inversions. Let $\mathbf{x}$ be the parameter vector of discretized values of the objective function $s(\mathbf{r},\omega)$, and $\mathbf{y}$ be the vector of discretized data of the output $\phi(\mathbf{r},\omega)$. Bayesian estimation theory provides an ability to incorporate *a priori* information of $\mathbf{x}$ into the inversion. The Bayesian solution of an inverse problem is the *a posteriori* probability density function (pdf) $p(\mathbf{x}\,|\,\mathbf{y})$, i.e., the conditional pdf of $\mathbf{x}$ given the data vector $\mathbf{y}$. It can be expressed as:

$$p(\mathbf{x}\,|\,\mathbf{y})=\frac{p(\mathbf{y}\,|\,\mathbf{x})p(\mathbf{x})}{p(\mathbf{y})}, \tag{12.11}$$

where $p(\mathbf{y}\,|\,\mathbf{x})$ is the conditional pdf of $\mathbf{y}$ given $\mathbf{x}$, reflecting the forward relation in an inverse problem, and $p(\mathbf{x})$ is the *a priori* probability of $\mathbf{x}$. If the theoretical relation between parameters and data is available, $p(\mathbf{y}\,|\,\mathbf{x})=p(\mathbf{y}-L\mathbf{x})$ with L being the integral operator in Eq. (12.4). $p(\mathbf{x})$ is often assumed as a Gaussian probability density function (Tarantola and Valette, 1982)

$$p(\mathbf{x})=const\cdot\exp(-\tfrac{1}{2}(\mathbf{x}-\mathbf{x}_0)^T C_0^{-1}(\mathbf{x}-\mathbf{x}_0)), \tag{12.12}$$

where $\mathbf{x}_0$ is a vector of expected values and C_0 is a covariance matrix which specifies the uncertainties of the inversion.

The probabilistic formulation of the least-squares inversion can be derived based on Eq. (12.11). We see that in addition to the deterministic theoretical relation (i.e., Eq. (12.4)) imposing constraints between the possible values of the parameters, the modification (i.e., multiple measurements in a probabilistic sense) of the parameters in the inversion procedure is controlled by *a priori* information like Gaussian probability density functions. This *a priori* information can provide a tolerance to control the trajectory of every state of the solution until converging to equilibrium. The tolerance is strongly related to the covariance matrix C_0, which is further used to estimate the uncertainties of inversion. It can efficiently bound the changes in magnitude of the parameters and provide stability to the inversion. Therefore, the statistical approach can significantly enhance the robustness of inversion in the presence of high noise levels and allow an analysis of the uncertainty in the results. However, it is still to be questioned that the *a priori* information can handle the nonuniqueness of the inversion because the missing frequency components will not be solved out from the band-limited data. Additional hard constraints need to be imposed. Moreover, Bayesian strategy, similar to other statistical strategies, is only a probability theorem-based mathematical approach applicable to all kinds of inverse problems. It does not define an inherently probabilistic mechanism by which it is possible for an Earth model to physically fit the data. We are often unclear how well the control data support the probability model and thus how far the latter may be trusted. If the a *priori* information about the parameters is weak, the corresponding variance will be large, or even infinite. In this case, the profitability obtained by the probability model will be reduced. Due to the nonuniqueness of the inversion, the uncertainty estimation is only limited to the available frequency components.

2.4. Ambiguous physical relationship

A major challenge in generalized geophysical inversions is the ambiguous nature of the physical relationships between parameters and data or between two different kinds of parameters. For example, a certain value of acoustic impedance corresponds to a wide range of porosity units. Figure 12.1 shows an experimental data-based crossplot of compressional velocity against porosity for 104 samples of carbonate-poor siliciclastics at 40 MPa effective stress (Vernik, 1994). It illustrates that the scatter of the data point distribution reaches up to about 15 porosity units at a certain value of compressional velocity. In this case, even if the velocities of both solid and fluid phases are estimated correctly, no optimal physical models can yield accurate porosity estimates. The deterministic theoretical model of Eq. (12.1) does not define a multi-value mapping for most geophysical processes. The statistical strategy mentioned previously can make the least-squares inverse procedure practical for an ill-posed inverse problem, but it does not explain the ambiguous physical relationship. This ambiguous nature implies that the scale of such a parameter as velocity is not matched to that of porosity in rocks; the statistical behavior of the physical system should be considered by incorporating an intrinsically probabilistic description into physical models; or the effects of other parameters should also be incorporated to narrow the ambiguity. It is beyond the scope of this paper to discuss these matters in detail. However, the problem remains as the subject of dispute in generalized geophysical inversions.

In this chapter, I take velocity-porosity datasets ($v(t)-\phi(t)$) as an example to demonstrate an areal approximation algorithm based on the reduced version of Eq. (12.2) to empirically model the ambiguous relationship with a scatter distribution of point-cloud data. Boiled down to one sentence: we pick up the optimal overall trend of data point-clouds with some nonlinear transform f, and then model the scatter distribution of data point-clouds with some wavelet operator $w(t)$. The method can be expressed as $v(t) = f(\phi(t), w(t), \lambda(t))$ where $\lambda(t)$ is a nonlinear factor that can adjust nonlinearly the function form of the equation into an appropriate shape that fits any practical dataset. The Caianiello neural network provides an optimization algorithm to iteratively adjust $\lambda(t)$ in adaptive response to lithologic variations vertically along a well log. I will discuss this algorithm in section 3.6. As a result, a joint lithologic inversion scheme is developed to extract porosity from acoustic velocity by first the inverse-operator-based inversion for initial model estimation and then the forward-operator-based reconstruction that improves the initial model.

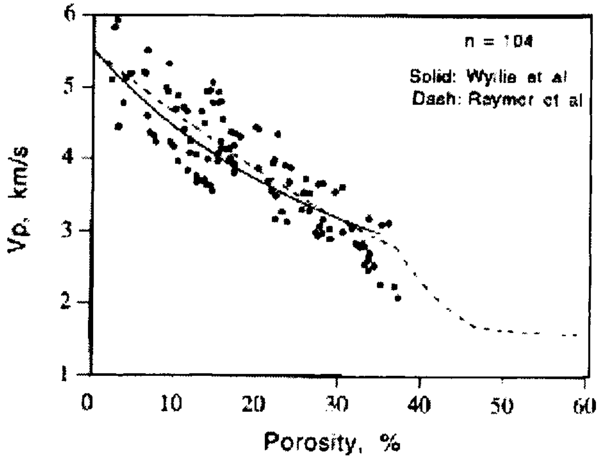


Figure 12.1. An experimental data-based crossplot of compressional velocity against porosity for 104 samples of carbonate-poor siliciclastics at 40 MPa effective stress. Note that the scatter of the data point distribution reaches up to about 15 porosity units at a certain value of compressional velocity. (From Vernik, 1994, used with the permission of *Geophysics*.)

3. CAIANIELLO NEURAL NETWORK METHOD

3.1. McCulloch-Pitts neuron model

Mathematically, the input-output relationship of a McCulloch-Pitts neuron is represented by inputs x_i, outputs x_j, connection weights w_{ji}, threshold θ_j, and differentiable activation function f as follows $x_j = f(\sum_{i=1}^{N} w_{ji}x_i - \theta_j)$. Due to the dot product of weights and inputs, the neuron outputs a single value when the input vector is a spatial pattern or a time signal. The model cannot process the frequency and phase information of an input signal. It is the connection mode among neurons that provides these neural networks with computational power.

3.2. Caianiello neuron model

The Caianiello neuron equation (Caianiello, 1961) is defined as

$$o_j(t) = f(\sum_{i=1}^{N} \int_0^t w_{ji}(\tau)o_i(t-\tau)d\tau - \theta_j(t)), \tag{12.13}$$

where the neuron's input, output, bias, and activation function are represented by $o_i(t)$, $o_j(t)$, $\theta_j(t)$, and f, respectively, and $w_{ji}(t)$ is the time-varying connection weight. The neuron equation (12.13) represents a neuron model with its spatial integration of inputs being a dot-product operation similar to the McCulloch-Pitts model, but with its temporal integration of inputs being a convolution. The weight kernel (a neural wavelet) in Eq. (12.13) is an information-detected operator used by a neuron. The input data will be convolution-stacked over a given interval called perceptual aperture, also referred to, in this paper, as the length of a neural wavelet. The perceptual aperture of the weight kernel, in general, is finitely large because the input data are detected only in a certain range. The location and size of the perceptual aperture affect the quality of information pick-up by the weight kernel. The aperture should correspond to the length of the weight function of a visual neuron. Based on numerous investigations of the visual system, the perceptual aperture is a fixed parameter, independent of the length of the input signal to the neuron and may have different values for neurons with different functions. This property determines local interconnections instead of global interconnections among neurons in a neural network. In practical applications, the weight kernel should be modified so it tapers the inputs near the boundary of the aperture.

The structure of the optimal perceptual aperture is strongly related to the spectrum property of the weight kernel, i.e., the amplitude-phase characteristics of the neural wavelet. Based on experimental results in vision research, the main spatiotemporal properties of major types of receptive fields in different levels of vertebrates may be described in terms of a family of extended Gabor functions (Marcelja, 1980; Daugman, 1980). That is, the optimal weight functions in equation (12.13) for a visual neuron are a set of Gabor basis functions that can provide a complete and exact representation of an arbitrary spatiotemporal signal. An example of the 1-D Gabor function is pictured in Figure 12.2. The neuron's filtering mechanism, intrinsically, is that its weight kernels crosscorrelate with the inputs from other neurons, and large correlation coefficients denote a good match between the input information and the neuron's filtering property. The neurons with similar temporal spectra gather to complete the same task using what are known as statistical population codes. For engineering applications, we replace the Riemann convolution over 0 to t in Eq. (12.13) with a conventional convolution integral over $-\infty$ to $+\infty$. The Caianiello neuron has been extended into a 4D filtering neuron to include spatial frequencies for both space- and time-varying signal processing (Fu, 1999c).

3.3. The Caianiello neuron-based multi-layer network

The architecture of a multi-layer network based on the Caianiello neuron is similar to the conventional multi-layer neural network (Rumelhart et al., 1986), except that each parameter becomes a time sequence instead of a constant value. Each neuron receives a number of time signals from other neurons and produces a single signal output that can fan out to other neurons. If the dataset used to train the neural network consists of an input matrix $o_i(t)$ ($i = 1,2,\ldots,I$, where I is the number of input time signals) and the desired output matrix $o_k(t)$ ($k = 1,2,\ldots,K$, where K is the number of output time signals), one can select an appropriate network architecture with I neurons in the input layer and K neurons in the output layer. For a general problem, one hidden layer between the input and output layers is enough. The mapping ability of the Caianiello neural network results mainly from the nonlinear activation function in Eq. (12.13). In general, the sigmoid nonlinearity of neurons is used. In Section 4.3, a physically meaningful transform will be described that can be used as the activation function for geophysical inversions.

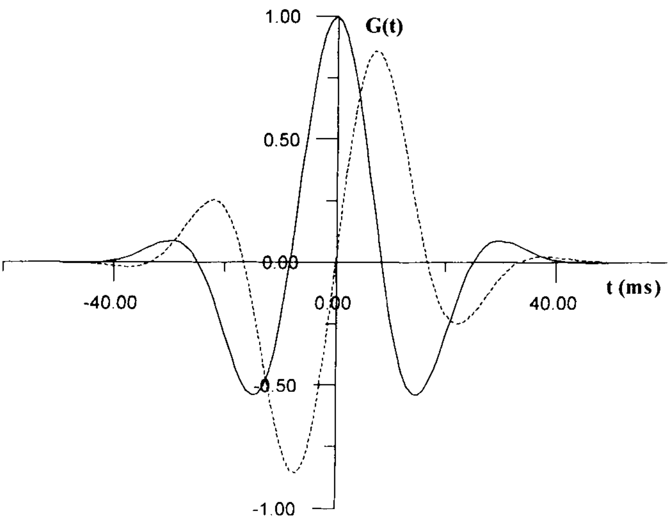


Figure 12.2. Examples of the one-dimensional Gabor function. Solid curve is the cosine-phase (or even-symmetric) version, and dashed curve is the sine-phase (or odd-symmetric) version.

3.4. Neural wavelet estimation

The neural wavelet of each neuron in the network can be adjusted iteratively to match the input signals and desired output signals. The cost function for this problem is the following mean-square error performance function

$$E=\frac{1}{2}\sum_k\sum_t e_k^{2}(t)=\frac{1}{2}\sum_k\sum_t[d_k(t)-o_k(t)]^2, \qquad (12.14)$$

where $d_k(t)$ is the desired output signal and $o_k(t)$ is the actual output signal from the output layer of the network. The application of the back-propagation technique to each layer leads to an update equation for neural wavelets in all neurons in this layer. The equation has a general recursion form for any neuron in any layer. For instance, from the hidden layer **J** down to the input layer **I**, the neural wavelet modification can be formulated as

$$\Delta w_{ji}(t)=\eta(t)\delta_j(t)\otimes o_i(t), \qquad (12.15)$$

where $\otimes$ is the crosscorrelation operation symbol and $\eta(t)$ is the learn rate which can be determined by automatic searching.

Two cases are considered to calculate the back-propagation error $\delta_j(t)$. For the output layer, the error $\delta_k(t)$ through the *k*th neuron in this layer is expressed as

$$\delta_k(t) = e_k(t) f'(net_k(t) - \theta_k(t)), \tag{12.16}$$

with

$$net_k(t) = \sum_j w_{kj}(t) * o_j(t), \tag{12.17}$$

where $*$ is the convolution operation symbol. For any hidden layer, $\delta_j(t)$ is obtained by the chain rule

$$\delta_j(t) = f'(net_j(t) - \theta_j(t)) \sum_k \delta_k(t) \otimes w_{kj}(t), \tag{12.18}$$

with

$$net_j(t) = \sum_i w_{ji}(t) * o_i(t). \tag{12.19}$$

The error back-propagation and the neural wavelet update use crosscorrelation operations while the forward propagation uses temporal convolution. A block frequency-domain implementation with FFTs for the forward and back-propagation can be used in the Caianiello network. There are two techniques for performing convolution (or correlation) using FFTs, known as the overlap-save and overlap-add sectioning methods (e.g., Robiner and Gold, 1975; Shynk, 1992). Frequency-domain operations have primarily two advantages compared to time-domain implementations. The first advantage is fast computational speed, provided by FFTs. A second advantage is that the FFT generates signals that are approximately uncorrelated (orthogonal). As a result, a time-varying learning rate can be used for each weight change, thereby allowing a more uniform convergence rate across the entire training. It has been recognized that the eigenvalue disparity of the input signal correlation matrix generally determines the convergence rate of a gradient-descent algorithm (Widrow and Stearns, 1985). These eigenvalues correspond roughly to the power of the signal spectrum at equally spaced frequency points around the unit circle. Therefore, it is possible to compensate for this power variation by using a learning rate (called the step size) that is inversely proportional to the power levels in the FFT frequency bins so as to improve the overall convergence rate of the algorithm (Sommen et al., 1987).

The information processing mechanism in the Caianiello network is related to the physical meanings of convolution and crosscorrelation. The adaptive adjustments of neural wavelets make the network adapt to an input information environment and perform learning tasks. The statistical population codes through large numbers of neurons with similar temporal spectrums in the network are adopted during the learning procedure and controlled by a physically meaningful transform f. The combination of the deterministic transforms and statistical population codes can enhance the coherency of information distribution among neurons, and, therefore, to infer some information lost in data or recover the information contaminated by noise.

3.5. Input signal reconstruction

In general, computational neural networks are used first through learning (weight adjustments) in an information environment known both for the inputs and the desired outputs. Once trained, they can be applied to any new input dataset in a new information environment with known inputs but unknown outputs. In many cases of geophysical inversion problems, we have known outputs but unknown or inexact inputs. Therefore, the new information environment also needs to be changed to adapt to the trained neural network.

Perturbing the inputs and observing the response of a neural network with the hope of achieving a better fit between the real and desired outputs will lead to a model-based algorithm for input signal reconstruction using neural networks. The forward calculations and cost function for this case are similar to those in section 3.4. We first consider the derivatives of E with respect to $o_j(t)$ input to the jth neuron in the hidden layer. The input signal modification in the hidden layer can be formulated as

$$\Delta o_j(t) = \eta(t)\delta_k(t) \otimes w_{kj}(t), \tag{12.20}$$

where the backprop error $\delta_k(t)$ through the kth layer is determined by Eq. (12.16). Likewise, defining the backprop error $\delta_j(t)$ through the jth layer as Eq. (12.18) leads to the update equation for $o_i(t)$ in the input layer:

$$\Delta o_i(t) = \eta(t)\delta_j(t) \otimes w_{ji}(t). \tag{12.21}$$

In comparison with the neural wavelet update scheme, we see that the back-propagation errors in both cases are the same. The crosscorrelation of these errors with the input signal to each layer leads to an update equation for the neural wavelets of neurons in this layer. Meanwhile, with the crosscorrelation of the back-propagation errors with the neural wavelets in each layer, we can obtain a recurrence formula to reconstruct the input signals of this layer. The convergence properties in both cases are almost the same. The method to reconstruct the input signal of the Caianiello network will be used to perform the forward-operator-based reconstruction for geophysical inverse problems.

3.6. Nonlinear factor optimization

As mentioned in Section 2.4, the adjustment of the time-varying nonlinear factor $\lambda(t)$ is needed for obtaining an optimal trend to fit point-cloud data. The application of the error-back-propagation technique to neurons of each layer yields an update equation for the nonlinear factors in this layer. Define the cost function for this problem as Eq. (12.14). The update equation for $\lambda(t)$ has a general recursion form for any neuron in any layer. For instance, the nonlinear factor modification for $\lambda_i(t)$ in the input layer can be expressed as

$$\Delta\lambda_i(t) = \beta(t) r_i(t) f'(\lambda_i(t)), \tag{12.22}$$

where $\beta(t)$ is the gain vector and the correlation function $r_i(t) = \sum_j \delta_j(t) \otimes w_{ji}(t)$ with $\delta_j(t)$ being Eq. (12.18).

4. INVERSION WITH SIMPLIFIED PHYSICAL MODELS

4.1. Simplified physical model

According to Sheriff (1991), a simplified model may be used to generate a catalog of master curves (or overall trends) for use in comparison with observed data. For instance, an exact seismic convolutional model for isotropic, perfectly elastic models of the earth can be expressed as Eq. (12.3), i.e., the convolution of a source signature with the impulse response of the earth. In the model, the key concept is linearity (Ziolkowski, 1991). It is well known that the inverse source problem is ill conditioned because important source spectral components may be suppressed in recording the data. Thus, the estimation of the physical source signature (source wavelet estimation) is generally based on two assumptions: band-limited source spectrum (matching to the bandwidth of the data) and point source excitation (leading to a far-field approximation). Using the statistical properties of the data for seismic wavelet estimation instead of source signature measurements leads to the well-known Robinson seismic convolution model. One object of wavelet estimations is to deconvolve the wavelet from reflection seismograms and recover the earth impulse response that, however, does not represent the subsurface reflection coefficients explicitly and uniquely. The computed earth impulse response is band-limited due to the band-limited seismic wavelet and contains all possible arrivals (reflection, refraction, multiples, and diffractions), noises, and transmission effects. The earth impulse response can be simplified as a time-slowness domain reflectivity by applying high-frequency asymptotics (Beylkin, 1985, Sacks and Symes 1987) to a family of one-dimensional equations for wave propagation in a layered medium (Treitel et al., 1982). It can be further reduced in a weak-contrast medium to the series of normal-incidence reflection coefficients for a single plane wave source at near-normal incidence (Lines and Treitel, 1984). The so called simplified Goupillaud earth model (1-D zero-offset model of the weak-contrast layered earth) has been often used to generate the zero offset reflection seismogram.

The argument among geophysicists regarding the Robinson seismic convolutional model is how to understand the seismic wavelet because of its ambiguity in physics. A reasonable physical interpretation is that the seismic wavelet is characterized by both source signature and transmission effects (Dobrin and Savit, 1988). The extra extension of the wavelet concept is based on the fact that wavelets that we can solve are always band-limited. This definition of seismic wavelet becomes practically significant because the effects of the seismic wavelet on seismograms are independent of the reflection coefficients of the earth, but rely on transmission and attenuation effects by its travel path. It is the changing wavelet model that I need in the joint inversion for representing the combined effects of source signature, transmission, and attenuation. Obviously, the successful application of the wavelet model is based on the fact that these effects are supposed to gradually vary laterally along seismic stratigraphic sequences. It is difficult to quantify elastic transmission effects and anelastic attenuation. From the bandwidth and dominant-frequency variations of seismic data, seismic wavelets generally vary vertically much more than laterally. High-quality seismic data often show that the variations of seismic waveform are changing rather gradually laterally along each large depositional unit associated with the blocky nature of the impedance distribution. The joint inversion makes use of this point by the elaborate implementation of an algorithm with stratigraphic constraints.

4.2. Joint impedance inversion method

Consider the following Robinson seismic convolutional model

$$x(t) = r(t) * b(t), \tag{12.23}$$

where $x(t)$ is the seismic trace, $r(t)$ the reflection coefficients and $b(t)$ the seismic wavelet which is thought of as an attenuated source wavelet. In general, solving $r(t)$ and $b(t)$ simultaneously is ill-posed from the equation. Minkoff and Symes (1995) showed that the band-limited wavelets and reflectivities could be estimated by simultaneous inversion if the rate of change of velocity with depth is sufficiently small. Harlan (1989) used an iterative algorithm to alternately estimate $r(t)$ and $b(t)$ in the offset domain by combining the modeling equations for hyperbolic traveltimes and convolutional wavelets. An analogous scheme was implemented in the time-slowness domain (Minkoff and Symes 1997). The realistic method for seismic wavelet estimation is the use of the well-derived, "exact" reflection coefficients (e.g., Nyman et al., 1987; Richard and Brac, 1988; Poggiagliolmi and Allred, 1994). For the integration of seismic and well data, I utilize the well-derived method for seismic wavelet estimation in this study.

In general, the deconvolution-based method (i.e., inverse-operator-based inversion) tends to broaden the bandwidth of seismic data with the purpose of obtaining a high-resolution result. The missing geological information, however, may not be recovered on the extended frequency band and the introduction of noise impairs the performance of these algorithms. For the model-based method (i.e., forward-operator-based inversion), the model space of the solution is reduced by the band-limited forward-operators, which can reduce the effect of noise on the solution. The resulting impedance model, however, is too smooth. The information that belongs to the null space cannot be solved, in principle, using the band-limited seismic data. Recovery of a portion of the information, especially in low- and high-frequencies, may only resort to well data and geological knowledge. This study presents a joint inversion scheme, i.e., combining both the model-based and deconvolution-based methods to integrate seismic data, well data, and geological knowledge for acoustic impedance.

There is a relatively large amount of information that is not completely absent from seismic data, but weak, incomplete, and distorted by noise. As is often true, the smooth impedance model estimated by some methods shows that this portion of information contained in seismic data is discarded during the inversion procedure. The reconstruction of this portion of information is a crucial target for various inversion methods, in which the elimination of noise is a critical procedure. The traditional inversion methods assume a deterministic forward relation for an impedance estimation problem. To overcome some disadvantages of the deterministic methods and also to exploit the statistical properties of the data, geostatistical techniques are becoming increasingly popular. These approaches can significantly enhance the robustness of inversion in the presence of high noise levels. Obviously, the successful application of these methods requires that the statistical relationship be constructed to cover a complicated reservoir system primarily described by deterministic theories. In this study, I add a statistical strategy (the Caianiello neural network) to the joint inversion in an attempt to combine both deterministic and statistical approaches to enhance the robustness of inversion in the presence of noise.

Neural networks solve a problem implicitly through network training with several different examples of solutions to the problem. Therefore, the examples selected as the solutions to the problem become very important to the problem, even more than the neural network itself. This mapping requires that the examples must be selected to describe the underlying physical relationship between the data. However, if the Caianiello network is related to the seismic convolutional model, it will reduce the harsh requirements for the examples used to train the network and meanwhile take advantage of the statistical population codes of the network. In the joint inversion, the neural wavelet estimation approach will be incorporated with the seismic convolutional model to estimate multistage seismic (MS) wavelets and multistage seismic inverse (MSI) wavelets.

In conclusion, The term "joint inversion" refers to (1) combining both the inverse-operator-based inversion and forward-operator-based inversion; (2) integrating seismic data, well data, and geological knowledge for impedance estimation; (3) incorporating the deterministic seismic convolutional model into the statistical neural network in the inversion.

4.3. Nonlinear transform

According to the seismic convolutional model (12.23) and the following recursive approximation between the acoustic impedance $z(t)$ and reflection coefficients (Foster, 1975)

$$r(t) \approx \frac{\partial \ln z(t)}{\partial t}, \qquad (12.24)$$

two kinds of simple forms of transform f can be obtained which will be used in the Caianiello neural network for the joint inversion. The first transform gives a mapping from the acoustic impedance $z(t)$ (as the input to the neural network) to the seismic trace $x(t)$ (as the output). Letting $\hat{z}(t) = \ln z(t)$, the seismic trace $x(t)$ can be expressed approximately by

$$x(t) = f\left[\hat{z}(t) * b(t)\right], \qquad (12.25)$$

where the activation function can be defined as the differential transform, $f = \partial/\partial t$, or alternatively, the linear transform $f(x) = x$ can be used with the replacement of $\hat{z}(t)$ by $r(t)$. Equation (12.25) can be decomposed into a multistage form, with each stage producing a filtered version of the subsurface logarithmic impedance $\hat{z}(t)$.

The second transform defines a nonlinear mapping from the seismic trace $x(t)$ (as the input to the neural network) to the acoustic impedance $z(t)$ (as the output). Letting $a(t)$ denote a seismic inverse wavelet, from the recursive relationship (12.24) the acoustic impedance $z(t)$ can be approximated as

$$z(t) = z_0 \exp\left[\int_0^t x(t) * a(t) dt\right]. \tag{12.26}$$

Define the exponential transform as

$$f(\cdot) = \exp\left[\int_0^t (\cdot) dt\right], \tag{12.27}$$

which can be further simplified (Berteussen and Ursin, 1983). With this substitution and letting the constant $z_0 = 1$, Eq. (12.26) becomes a standard form

$$z(t) = f[x(t) * a(t)]. \tag{12.28}$$

4.4. Joint inversion step 1: MSI and MS wavelet extraction at the wells

The algorithm scheme for neural wavelet estimation, combined with Eq. (12.28), is used to extract the MSI wavelets. The total training set consists of an input matrix $x_{il}(t)$ ($l = 1,2,\ldots,L$, where L is the number of wells in the interesting area; $i = 1,2,\ldots,I$, where I is the number of input seismic traces at the lth well, also denoting the number of neurons in the input layer), and a desired output matrix $z_{kl}(t)$ ($l = 1,2,\ldots,L$; $k = 1,2,\ldots,K$, where K is the number of impedance logs and the relevant signals associated with the lth well, also representing the number of neurons in the output layer). The Caianiello neural network is illustrated in Figure 12.3. In general, the parameter I is chosen large enough in the vicinity of each well to take advantage of the spatial correlation property among adjacent traces. The main difference of the network training procedure from regular applications of neural networks is that the direction and size of the weight adjustment that is made during each back-propagation cycle are controlled by Eq. (12.28) as an underlying physical relationship.

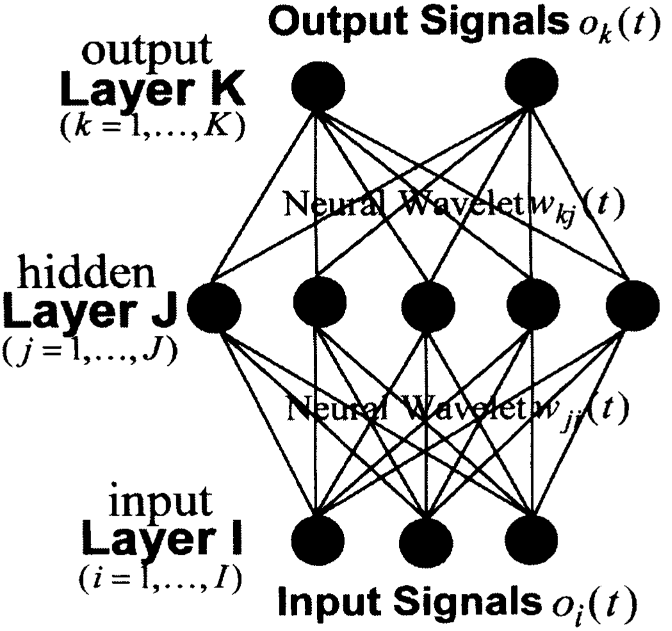


Figure 12.3. Three-layer Caianiello neural network architecture.

Once trained for all wells, one can have a neural network system for a seismic section or an interesting region, which, to some degree, gives a representation of the relationship of seismic data (as inputs) and the subsurface impedance (as outputs). The effects of multi-well data as a broadband constraint on the joint inversion are implicitly merged into the network by transformation of the MSI wavelets. Obviously, the information representation can be sufficient and reliable if more wells are available. For laterally stable depositional units, seismic wavelets are less laterally variable and sparse well control is also applicable. Nevertheless, the neural network system can be gradually improved by feeding new well data into it during the lifetime of an oil field.

Likewise, the neural wavelet estimation scheme is combined with Eq. (12.25) to perform the MS wavelet extraction. In contrast to the MSI wavelet estimation, here, the impedance log of each well is used as the input to the Caianiello neural network, and the seismic traces at this well are used as the desired output. The neural network training can be done by iteratively perturbing neural wavelets with hope of achieving a better fit between the seismic data and the well log derived synthetic data as the actual output of the network. It should be stressed that the MS wavelets are band-limited, matched to the seismic data. The extracted MS wavelets for all wells in an interesting area are stored in one Caianiello neural network in the form of its neural wavelets, which can be used to model seismic data directly from the impedance model. The information representation of the network can be easily refined by updating the existing network to honor new wells. Clearly the MS wavelet extraction algorithm is a model-based wavelet estimation procedure. It is important to realize that the model-based wavelet estimation is different from the model-based impedance inversion. For the former, what we need is a band-limited seismic wavelet matched to the seismic data spectra, rather than the latter where we need to figure out a broadband impedance model. It is straightforward from Eq. (12.25) to show that the MS wavelets only cover the source signature and its transmission and attenuation effects. The effects that the MS wavelets include have finally been left in seismic data. This is the basis for the joint inversion step 3.

4.5. Joint inversion step 2: initial impedance model estimation

The trained neural network with the MSI wavelets can then be used for the deconvolution to estimate initial impedance model away from the wells. In this processing, the network, established during the training phase, remains unchanged and seismic traces are now fed successively into the network and deconvolved at the outputs. This deconvolution method is a direct inversion method that attempts to estimate the impedance directly from seismic data. During the extrapolation phase, a set of new MSI wavelets can be autonomously produced by means of automatic interpolation of the network for the deconvolution of individual seismic trace between wells. The MSI wavelets, in a static manner different from dynamic iteration, approach the solution stage by stage in the deconvolution procedure. In addition, with the MSI wavelets, noise is broken down and scattered over many neurons so that the statistical population codes of the Caianiello network can increase the robustness of the deconvolution-based inversion in the presence of noise.

The estimated MSI wavelets are thought to be accurate at the wells, but may not be accurate for the seismic traces away from the wells. The errors in such MSI wavelets are transferred to the estimated impedance after deconvolution. This is the reason that the joint inversion step 3 below is needed to improve the estimated initial impedance. Let's further investigate this problem. The information contained implicitly in the MSI wavelets consists of two parts: the missing geological information and the effect of seismic wavelets. The latter is expected to be less varied laterally away from the wells, especially in the dominant frequency. This is often true for many of stationary depositional units. The first part, previously obtained from well logs, is used as the particular solutions at the wells with which the MSI wavelets may infer some missing information between wells to provide adequate information compensation for individual traces.

4.6. Joint inversion step 3: model-based impedance improvement

The trained neural network with the MS wavelets is used for the model-based inversion away from the wells to produce a final impedance profile. The purpose of this step is to improve the middle-frequency components in the initial impedance model. Here, seismic traces are used as the desired output of the network, and the initial impedance model obtained in step 2 is used as the input. The algorithm in this step is from the combination of the Caianiello-network-based input signal reconstruction scheme and Eq. (12.25). Similarly, for each trace to be inverted, a number of seismic traces around this trace can be employed to compose its desired output matrix. The following basic aspects are considered for this step.

Two major disadvantages have been acknowledged to be inherent in the model-based inversion algorithms. One is severe nonuniqueness caused by the band-limited seismic data and wavelets. Another is that the guess of the initial solution has a large influence on the convergence of the algorithm used. The deconvolution-based initial impedance estimation in step 2 assures the solution of the above two problems to a large degree. As mentioned in step 2, the MSI wavelets used for the deconvolution-based initial impedance inversion cover both the seismic wavelet effect and the missing geological information. Thus, the inversion in step 2 focuses on removing the seismic wavelet from the data, improving signal-to-noise ratio, and providing adequate high- and low-frequency information compensation for the trace to be inverted. The conversion efficiency of the middle-frequency information may not be perfect from reflection amplitude to acoustic impedance. The local distortions left in phase and amplitude need to be minimized further. In this step, the MS wavelets only account for the band-limited seismic wavelet. To use seismic data to their full extent, the robust model-based inversion with the MS wavelets is employed to further improve the middle-frequency components in the initial impedance model that are matched to the frequency band of the MS wavelets. In this situation, the solution is approached both step-by-step through dynamic iterations from an initial solution and stage-by-stage with a static representation of the MS wavelets.

For the information completely absent from seismic data, it may be inferred by the MSI wavelets according to the corresponding frequency components obtained from impedance logs of wells. This procedure is performed through the Caianiello network in step 2 to provide adequate information compensation for the individual trace away from the wells. In this step, these components in the initial impedance do not require updating since there is no corresponding information in the seismic data. The block frequency-domain implementations of the algorithm not only substantially reduce the computational complexity, but also enable a precise control of different frequency components to be inverted.

4.7. Large-scale stratigraphic constraint

It should be stressed that the lateral variations of the MS and MSI wavelets are assumed to be gradual from one well to another in each large depositional unit associated with the blocky nature of impedance distribution. Each such distinct zone of deposition has a family of the MS and MSI wavelets to represent its seismic behavior and geological properties. The lateral variations of the wavelets are mainly on the dominant frequency, because it generally has the largest effect on the inversion result among all relevant parameters. In fact, the dominant frequency and bandwidth of seismic data are less varied laterally than vertically. In the areas of complex geologic structures, such as faults with large throws, pinchouts, and sharp dips, specified stratal geometry to control main events should be specified as a stratigraphic constraint during the extrapolation in the joint inversion. This constraint ensures that the applications of the MS and MSI wavelets along the seismic line are laterally restricted within the same large-scale geological unit from which they are extracted at the wells, and change with geological structures, especially with large-throw faults. The stratal geometry is determined as follows: First a geological interpretation of the seismic section studied is conducted under well data control, determining the spatial distributions of some main geological units. Next a polynomial fitting technique is used to track main events and build reliable stratal geometry.

5. INVERSION WITH EMPIRICALLY-DERIVED MODELS

5.1. Empirically derived petrophysical model for the trend

For detailed understanding of the relationship between reservoir acoustic properties and porosity and/or clay content, the Raymer's equation was proposed as a modification of the Wyllie's time-average equation by suggesting different laws for different porosity ranges (Nur et al., 1998). The two models appear adequate for clay-free sandstones, but fail to describe shaly sandstones. Numerous advances are involved with the combined effects of porosity and clay content on acoustic properties. It is noteworthy that the Han's linear relation (Han et al., 1986) fits laboratory data quite well for a variety of lithologies in a relatively wide range of porosity. This suggests that empirically derived multivariate linear regression equations can be constructed by relating acoustic velocities to porosity and clay-content. Considering the lithologic inversion in complex geological environments, an empirically-derived and relatively flexible model is presented here with the intention to fit well log data for unconsolidated sandstone reservoirs in the complex continental sediments in western China,

$$\frac{\phi_m(t)(\phi_m(t)-2\phi(t))}{\phi(t)(\phi_m(t)-\phi(t))}=\lambda(t)\ln\left(\frac{v_p(t)-v_f(t)}{v_m(t)-v_p(t)}\right), \tag{12.29}$$

where $\phi(t)$ and $v_p(t)$ are the porosity and P-wave velocity curves in vertical time, respectively; $\phi_m(t)$, $v_m(t)$, and $v_f(t)$ are the maximum sandstone porosity, rock matrix velocity, and pore fluid velocity in the reservoir under study, respectively; and $\lambda(t)$ is a nonlinear factor that adjusts the function form to fit practical data points and can be optimally estimated by the Caianiello neural network method in section 3.6. One can estimate the $\phi_m(t)$, $v_m(t)$, and $v_f(t)$ values for various lithologies and fluids to match practically any dataset in complex deposits. The accurate estimation of the time-varying nonlinear factor $\lambda(t)$ for different lithologies at different depths is a crucial point in the applications of the model to the joint lithologic inversion. Similarly, some simple deterministic relationship between acoustic velocities and clay-content for clay-rich sandstones can be empirically derived (Fu, 1999a).

Several aspects are considered for the construction of Eq. (12.29) and its applications (Fu, 1999a). Neff's petrophysical-based forward modeling (Neff, 1990a,b) demonstrates the effects of changes in petrophysical properties (porosity, shale volume, and saturation) on seismic waveform responses, indicating that the petrophysical properties of reservoir units are highly variable vertically and horizontally. Accurate porosity estimate and lithology prediction from acoustic velocities need the determination of petrophysical relationships to be based on the detailed petrophysical classification (Vernik and Nur, 1992; Vernik, 1994). In my papers (Fu, 1999b), I took the case proposed by Burge and Neff (1998) as an example to demonstrate the performance of Eq. (12.29), which illustrates the distinct variation in the impedance versus porosity relationship due to the lithologic variation and the change in fluid type of gas condensate versus water within the siliciclastic unit, each distinct lithologic unit having a unique set of petrophysical constants and equations. As a result, the rule from Eq. (12.29) can also describe the impedance-porosity relationships for different lithologic units. This indicates that Eq. (12.29) may provide a possible means to facilitate implementation of the petrophysical classification scheme for practical lithologic inversion.

A class of functions similar to Eq. (12.29) and their evolving versions has been widely applied to describe a physical process with stronger state variations occurring in the early and late stages than in the middle. This physical phenomenon widely exists in the natural world. This implies a local sudden change occurring in the process. In fact, numerous experimental data from rock physics laboratories suggest that there exists a critical porosity that separates the entire porosity range (from 0 ~ 100%) into different porosity domains with different velocity-porosity behavior (Nur, 1992; Nur et al., 1998). This critical porosity becomes a key to relating acoustic properties to porosity for the reservoir interval with a remarkably wide range of porosity distribution. The nonlinear transform of Eq. (12.29) is constructed with an attempt to apply the critical porosity concept to the joint lithologic inversion.

5.2. Neural wavelets for scatter distribution

Even if the deterministic petrophysical model is calculated optimally, it provides only a trend to fit data points on a scatterplot. The trend is one side of the relationship of acoustic properties to porosity. Another is the scatter distribution of data point-clouds around the trend. The scatter distribution could be referred to as the trend's receptive field, the range to which the influence of the trend can reach. This scattering behavior has drawn much interest recently, motivated by its role to transform acoustic properties into porosity. I crosscorrelate a scanning operator with porosity curves to quantify the deviations of data points from the trend for each lithology. Neural wavelets in the Caianiello neural network provide an effective means to facilitate the implementation of this strategy. The use of neural wavelets cannot narrow the deviations of data points from the trend unless other seismic attributes are incorporated but can capture the deviations with a boundary of arbitrary shape to make a distinction between different lithologies. This is actually an integration of the neural network-based pattern classification with deterministic velocity-porosity equations, which can provide an areal approximation to velocity-porosity datasets. Especially in the case of shale, the pattern classification will be dominant in the procedure of lithologic simulation. The aperture of the neural wavelet depends on the range of scatter distribution of data points. Sandstones containing different amounts of volume clay have different regions of scatter distributions of data points in the velocity-porosity space as well as different deviation levels from the trends, which correspond to different apertures and spectral contents of neural wavelets.

5.3. Joint inversion strategy

The Caianiello neural network method (including neural wavelet estimation, input signal reconstruction, and nonlinear factor optimization) is incorporated with the deterministic petrophysical models into a joint lithologic inversion for porosity estimation. First, a large number of well-data-based numerical modelings on the relationships of acoustic impedance and porosity are needed to determine cutoff parameters. Second, neural wavelets are used as scanning operators to discern data-point scatter distributions and separate different lithologies in the impedance-porosity space.

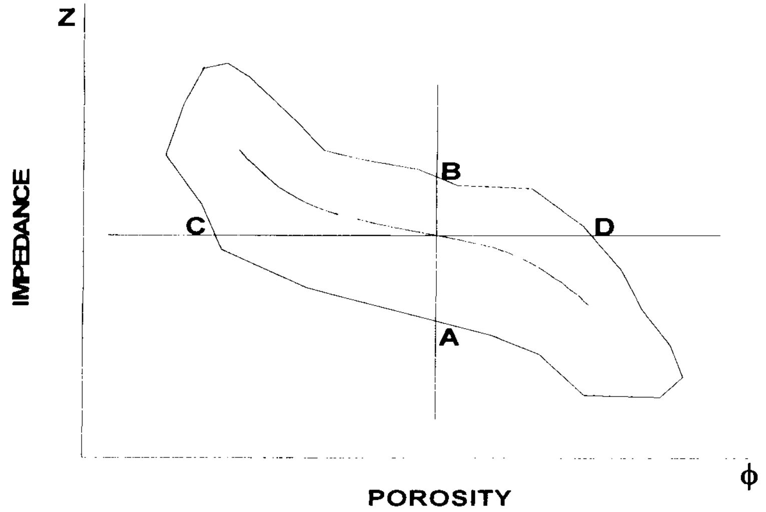


Figure 12.4. Schematic description of the joint lithologic inversion. First a deterministic petrophysical model defines the overall trend across the cloud of data points. Next neural wavelets determine the scatter distribution of data points around the trend curve along both the ϕ -axis (e.g., Line **CD**) and the z -axis (e.g., Line **AB**). (Reproduced with permission from Fu, 1999b.)

The joint lithologic inversion scheme consists of two subprocesses. First, inverse neural wavelets are extracted at the wells, and then the inverse-operator-based inversion is used to estimate an initial porosity model away from the wells. This can be expressed as: $\phi(t) = f(z(t), w_z(t), \lambda(t))$ where the deterministic petrophysical model f together with its nonlinear factor $\lambda(t)$ and cutoff parameters can define trend curves, and the crosscorrelation of the impedance $z(t)$ with the inverse neural wavelets $w_z(t)$ can determine the data-point scatters around the trend curve in the direction of the z -axis (e.g., Line **AB** in Figure 12.4). It should be mentioned that the statistical population codes of numerous neurons in the Caianiello network are used in this procedure. Second, forward neural wavelets are estimated at the wells, and then the forward-operator-based reconstruction is employed to improve the porosity model. This can be expressed as $z(t) = f(\phi(t), w_\phi(t), \lambda(t))$. The crosscorrelation of the porosity $\phi(t)$ with the forward neural wavelets $w_\phi(t)$ can evaluate the deviations from the trend along the lines parallel to the ϕ -axis (e.g., Line **CD**).

6. EXAMPLE

The joint inversion described above has been applied to acoustic impedance, porosity, and clay-content estimations in several oil fields of China. In this section, I will show an example to demonstrate the performance of the joint inversion scheme for acoustic impedance estimation in a clastic field. The seismic data in Figure 12.5 crosses two wells. The data show heterogeneous properties of the continental deposits. The interesting zone with a number of reservoir distributions is located at the delta front facies deposited with sandstone-mudstone sequence. Integration of multi-well information consistently and reasonably in an impedance inversion is particularly challenging. In the joint inversion, the MSI and MS wavelets for all wells are simultaneously extracted at the wells and stored in the form of neural wavelets. This implies the inversion is based on a reasonable starting point to recover information. For an individual seismic trace between wells, the neural network can autonomously develop a set of appropriate MSI and MS wavelets in adaptive response to this trace. In this way, the trace is inverted consistently from one well to another. Inversions of the data, under the control of these two wells, are demonstrated in Figure 12.6. The well-derived impedance logs of these two wells are inserted at the wells on the impedance profile so that one can track and correlate layers. The right part of the profile is a productive area with two major oil-bearing sand layers located respectively at about 2300 ms and 2500 ms (marked with arrows), which, however, are getting poor toward the left and become only oil-bearing indication on the left well. Two large fault zones exist in between. The purpose of inversion is to track and correlate lateral variations of the reservoir from the right to the left. The changes in reservoir thickness and relative quality in the estimated impedance confirm the geological interpretation based on the wells. These results significantly improve the spatial description of reservoirs.

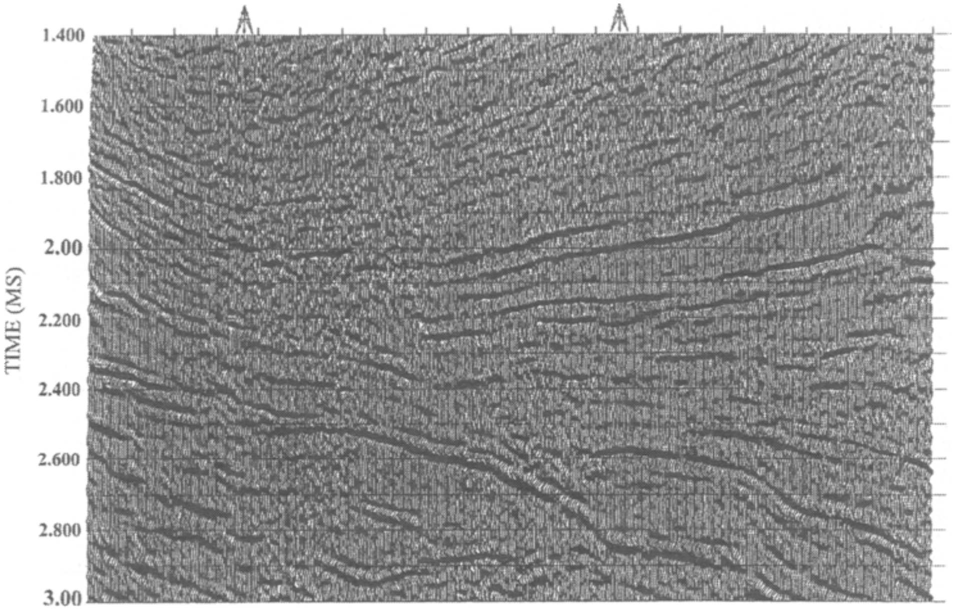


Figure 12.5. A seismic section corresponding to a continental clastic deposit.

Since the impedance section can map individual lithologic units, including both the physical shape of the unit and lateral variations in lithology, the most useful feature of the section lies in the fact that the reservoir characterization that results from the wells can be directly extended away from the wells via the impedance variations of individual lithologic units. It should be stressed that high-fidelity impedance sections depend on relative amplitude preservation of seismic data. Specifically, linear noise can be removed in the joint inversion as long as those of the traces at the wells account for its underlying noise mechanism. Random noise can be, to a large extent, minimized by the neural network approach used in the joint inversion. Multiple reflections have bad influences on the estimated impedance if they are strong and smearing the reflection zone of interest. In general, interbed multiples are relatively weak in the area of sandstone-mudstone sequence deposition. Amplitude distortions usually lead to the fact that some frequency components of seismic data are absent, incomplete, or incorrect. As mentioned before, if the amplitude distortion is not individual, but distributed over many adjacent seismic traces, it will severely impair the estimated impedance. Consequently, it is not easy to quantitatively measure the lateral variations away from the wells in the estimated impedance profile. However, these variations can basically reflect the relative changes in the real impedance model.

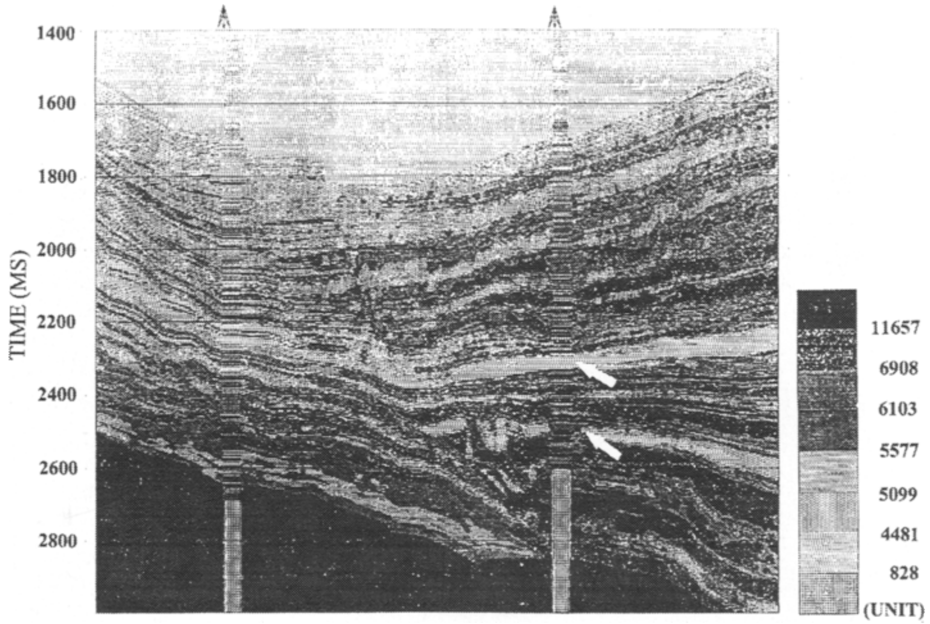


Figure 12.6. Impedance estimate guided by two wells. The borehole impedance logs of these two wells are plotted at the wells, respectively. (After Fu, 1997.)

7. DISCUSSIONS AND CONCLUSIONS

The Caianiello neuron model is used to construct a new neural network for time-varying signal processing. The Caianiello neural network method includes neural wavelet estimation, input signal reconstruction, and nonlinear factor optimization. Some simplified theoretical relationships or empirically derived physical models, relating subsurface physical parameters to observed geophysical data, can be introduced into the Caianiello neural network via nonlinear activation functions of neurons. The combination of the deterministic physical models and the statistical Caianiello network leads to an information integrated approach for geophysical inverse problems. As a result, a new joint inversion scheme for acoustic impedance and lithologic estimations has been built by integrating broadband seismic data, well data, and geological knowledge. The main conclusions can be summarized as follows:

1) Geophysical inversion is a procedure of information recovery as well as multidisciplinary information integration. Geophysical inverse problems almost always lack uniqueness, stability, and certainty. Due to a limited amount of observed data from each discipline, information recovery by inversion has to resort to integration of data from different sources. Ambiguous physical relationships, relating observed geophysical data to subsurface physical properties, suggest that geophysical inverse problems be characterized by both deterministic mechanism and statistical behavior. Therefore, the optimal inversion method is the one with the ability to aptly merge certain deterministic physical mechanisms into a statistical algorithm.
2) For acoustic impedance estimation, the Robinson seismic convolutional model is used to provide a physical relationship for the Caianiello neural network. Considering the complexity of the subsurface media, the seismic wavelet is often thought of as an attenuated source wavelet, characterized by source signature, transmission, and attenuation effects. According to information theory, the Robinson seismic convolutional model is irreversible due to the band-limited seismic wavelet. The seismic inverse wavelet, if needed, has a completely different content in terms of information conservation. That is, the seismic inverse wavelet not only accounts for the effect of the seismic wavelet, but also more importantly, contains the missing geological information. In this sense, a combined application of the seismic wavelet and seismic inverse wavelet can produce optimal impedance estimation.
3) For the inversion of porosity, the scatter distribution of the velocity-porosity data points indicates that rocks with different lithologic components have three different properties: (a) the different shape of trends that imply the relationship of velocity to porosity, (b) the different location of datapoint distribution in the velocity-porosity space, and (c) the different deviation extent of datapoint scatterings from the trend. Any lithologic inversion method should take these three aspects into account. In this chapter, I give an empirically derived, relatively flexible petrophysical model relating acoustic velocities to porosity for clay-bearing sandstone reservoirs. It is based on the facts that the different porosity ranges have different gradients of trends. The deterministic petrophysical model can be used as the nonlinear activation function in the Caianiello neural network for porosity estimation. This is actually an integration of the deterministic petrophysical relationship with the neural network-based pattern classification, the former for picking up the trends of different lithologic units and the latter for quantifying datapoint deviations from the trends to distinguish among different lithologic units in the data space.
4) The joint impedance inversion consists of two processes. First, seismic inverse wavelets are estimated at the wells, and then the inverse-operator-based inversion is used for initial impedance estimation to remove the effect of seismic wavelets and provide adequate high- and low-frequency information. Second, seismic wavelets are

extracted at the wells, and then, the forward-operator-based reconstruction can improve the initial impedance model to minimize local distortions left in phase and amplitude. To develop information representation of the seismic wavelet and seismic inverse wavelet, the Caianiello neural network provides an efficient approach to decompose these two kinds of wavelets into multistage versions. This multistage decomposition provides the joint inversion with an ability to approach the solution stage by stage in a static manner, increasing the robustness of the inversion.

5) The joint lithologic inversion consists of three processes. First, to pick up trends for any practical datasets in the velocity-porosity crossplot, we need to do lots of well-data-based numerical modelings to determine the cutoff parameters for different lithologies and fluids. Second, inverse neural wavelets are extracted at the wells to quantify the datapoint deviation from the trend along the velocity-axis, and then the inverse-operator-based inversion is used to estimate an initial porosity model away from the wells. Third, forward neural wavelets are estimated at the wells to quantify the datapoint deviation from the trend along the porosity-axis, and then the forward-operator-based reconstruction is implemented to improve the initial porosity model. The use of neural wavelets cannot narrow the deviation of data points from the trend. If appropriate petrophysical models are available, the incorporation of seismic waveform information into the joint lithologic inversion will allow for more accurate porosity estimate than only using velocity information.
6) For each trace between wells, a set of wavelets will be automatically interpolated by the Caianiello network based on those at the wells. The lateral variations (dominant frequency and bandwidth) of the wavelets are assumed to be gradual from one well to another in each large depositional unit associated with the blocky nature of impedance distribution. Each such distinct sediment zone has a family of wavelets to represent its petrophysical property and seismic characteristics. In the areas of complex geological structures, a specified, large-scale strata geometry to control main reflectors should be used as a stratigraphic constraint to ensure that the applications of wavelets are laterally restricted inside the same seismic stratigraphy unit from which they are extracted at the wells.
7) The frequency-domain implementation of the joint inversion scheme enables precise control of the inversion on different frequency scales. This makes it convenient to understand reservoir behavior on different resolution scales.

REFERENCES

Berteussen, K., and Ursin, B., 1983, Approximate computation of the acoustic impedance from seismic data: Geophysics, **48**, 1351-1358.

Beylkin, G., 1985, Imaging of discontinuities in the inverse scattering problem by inversion of a causal generalized radon transform: J. Math. Phys., **26**, 99-108.

Burge, D., and Neff, D., 1998, Well-based seismic lithology inversion for porosity and pay-thickness mapping: The Leading Edge of Exploration, **17**, 166-171.

Caianniello, E., 1961, Outline of a theory of thought-processes and thinking machines: J. Theoret. Biol., **2**, 204-235.

Daugman, J., 1980, Two-dimensional spectral analysis of cortical receptive field profiles: Vision Res., **20**, 847-856.

Dobrin, M., and Savit, C., 1988, Introduction to Geophysical Prospecting: 4th ed., McGraw-Hill.

Foster, M., 1975, Transmission effects in the continuous one-dimensional seismic model: Geophys. J. Roy. Astr. Soc., **42**, 519-527.

Fu, L., 1995, An artificial neural network theory and its application to seismic data processing: PhD thesis, University of Petroleum, Beijing, PRC.

Fu, L., 1997, Application of the Caianiello neuron-based network to joint inversion: 67th Ann. Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts, 1624-1627.

Fu, L., 1998, Joint inversion for acoustic impedance: Submitted to Geophysics.

Fu, L., 1999a, An information integrated approach for reservoir characterization, *in* Sandham, W., and Leggett, M. Eds., Geophysical Applications of Artificial Neural Networks and Fuzzy Logic: Kluwer Academic Publishers, in press.

Fu, L., 1999b, Looking for links between deterministic and statistical methods for porosity and clay-content estimation: 69th Ann. Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts.

Fu, L., 1999c, A neuron filtering model and its neural network for space- and time-varying signal processing: Third International Conference on Cognitive and Neural systems, Boston University, Paper Vision B03.

Fu, L., Chen, S., and Duan, Y., 1997, ANNLOG technique for seismic wave impedance inversion and its application effect: Oil Geophysical Prospecting: **32**, 34-44.

Han, D., Nur, A., and Morgan, D., 1986, Effects of porosity and clay content on wave velocities in sandstones: Geophysics, **51**, 2093-2107.

Harlan, W., 1989, Simultaneous velocity filtering of hyperbolic reflections and balancing of offset-dependent wavelets: Geophysics, **54**, 1455-1465.

Lines, L., and Treitel, S., 1984, A review of least square inversion and its application to geophysical problems: Geophys. Prosp., **32**, 159-186.

Marcelja, S., 1980, Mathematical description of the responses of simple cortical cells: J. Opt. Soc. Am., **70**, 1297-1300.

McCulloch, W., and Pitts, W., 1943, A logical calculus of the ideas immanent in nervous activity: Bull. of Math. Bio., **5**, 115-133.

Minkoff, S., and Symes, W., 1995, Estimating the energy source and reflectivity by seismic inversion: Inverse Problems, **11**, 383-395.

Minkoff, S., and Symes, W., 1997, Full waveform inversion of marine reflection data in the plane-wave domain: Geophysics, **62**, 540-553.

Neff, D., 1990a, Incremental pay thickness modeling of hydrocarbon reservoirs: Geophysics, **55**, 558-566.

Neff, D., 1990b, Estimated pay mapping using three-dimensional seismic data and incremental pay thickness modeling: **55**, 567-575.

Nur, A., 1992, The role of critical porosity in the physical response of rocks: EOS, Trans. AGU, **43**, 66.

Nur, A., Mavko, G., Dvorkin, J., and Galmudi, D., 1998, Critical porosity: A key to relating physical properties to porosity in rocks: The Leading Edge of Exploration, **17**, 357-362.

Nyman, D., Parry, M., and Knight, R., 1987, Seismic wavelet estimation using well control: 57th Ann. Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts, 211-213.

Poggiagliolmi, E., and Allred, R., 1994, Detailed reservoir definition by integration of well and 3-D seismic data using space adaptive wavelet processing: The Leading Edge of Exploration, **13**, No. 7, 749-754.

Richard, V., and Brac, J., 1988, Wavelet Analysis using well-log information: 58th Ann. Internat. Mtg., Soc. Expl. Geophys., Expanded Abstracts, 946-949.

Robiner, L., and Gold, B., 1975, Theory and Application of Digital Signal Processing: Prentice-Hall.

Robinson, E., 1954, Predictive decomposition of time series with application to seismic exploration: reprinted in Geophysics, 1967, **32**, 418-484.

Robinson, E., 1957, Predictive decomposition of seismic traces: Geophysics, **22**, 767-778.

Robinson, E., and Treitel, S., 1980, Geophysical Signal Analysis: Prentice-Hall, Inc.

Rumelhart, D., Hinton, G., and Williams, R., 1986, Learning representations by error propagation, *in* Rumelhart, D. E. and McClelland, J. L., Eds., Parallel Distributed Processing: MIT Press, 318-362.

Sacks, P., and Symes, W., 1987, Recovery of the elastic parameters of a layered half-space: Geophys. J. Roy. Astr. Soc., **88**, 593-620.

Sheriff, R., 1991, Encyclopedic Dictionary of Exploration Geophysics, 3rd Ed.: Soc. Expl. Geophys.

Shynk, J., 1992, Frequency-domain and multirate adaptive filtering: IEEE ASSP Magazine, **9**, 14-37.

Sommen, P., Van Gerwen, P., Kotmans, H., and Janssen, A., 1987, Convergence analysis of a frequency-domain adaptive filter with exponential power averaging and generalized window function: IEEE Trans. Circuits Systems, **CAS-34**, 788-798.

Treitel, S., Gutowski P., and Wagner, D., 1982, Plane-wave decomposition of seismograms: Geophysics, **47**, 1375-1401.

Tarantola, A., and Valette, B., 1982, Inverse problems: Quest for information: J. Geophys., **50**, 159-170.

Vernik, L., 1994, Predicting lithology and transport properties from acoustic velocities based on petrophysical classification of siliciclastics: Geophysics, **63**, 420-427.

Vernik, L., and Nur, A., 1992, Petrophysical classification of siliciclastics for lithology and porosity prediction from seismic velocities: AAPG Bull., **76**, 1295-1309.

Widrow, B., and Stearns, S. D., 1985, Adaptive Signal Processing: Prentice-Hall.

Ziolkowski, A., 1991, Why don't we measure seismic signatures?: Geophysics, **56**, 190-201.

This Page Intentionally Left Blank

Part III
Non-Seismic Applications

The third section of this book reviews applications of computational neural networks to surface and borehole data for potential fields, electromagnetic, and electrical methods. Chapter 13 reviews many published applications of computational neural networks for a variety of surveys. Chapter 14 details the application of neural networks to the interpretation of airborne electromagnetic data. A modified MLP architecture is used to process the airborne data and produce a 1D interpretation. Chapter 15 compares several network learning algorithms, previously described in Chapter 5, for a boundary detection problem with unfocused resistivity logging tools. Chapter 16 compares an RBF network to least-squares inversion for a frequency-domain surface electromagnetic survey. The network produced nearly identical results to the inversion but in a fraction of the time. Chapter 17 develops a method to assign a confidence factor to a neural network output for a time-domain data inversion. The network estimates values for the Cole-Cole parameters and a second network estimates the range of the error associated with the estimate in 5% increments.

With the exception of well logging applications and UXO surveys, neural network interpretation has not been commercialized or routinely used for non-seismic data interpretation. This is not surprising since software packages for non-seismic techniques do not have the same market potential as the seismic processing packages. Many of the applications developed by university researchers demonstrate a proof of concept but the technology has not been transferred to industry. While non-seismic geophysical interpretation software using neural networks may not be available anytime soon, I do believe more and more contractors will begin to integrate the technology, where appropriate, in their interpretations.

The neural network applications in Part II tend to focus on classification problems while the applications in Part III emphasize function estimation. This follows the trend in the literature, especially for the surface techniques, where the emphasis has been on estimating model parameters. Calderon et al.[1] shows that neural networks can outperform a least-squares inversion for resistivity data. The limitation in widely applying neural networks for inversion is the huge number of models that must be generated for training if the network is to be applied to all field surveys. The alternative is to create customized networks for different types of field situations. Classification problems, however, could be trained with fewer models or with field data. Applications that involve monitoring for changes in fluid movement or properties, changes in rock type or conditions during excavation, or anomaly detection are ideal classification problems for a neural network.

---

[1] Calderon-Macias, C., Sen, M., and Stoffa, P., 2000, Artificial neural networks for parameter estimation in geophysics: Geophysical Prospecting, **48**, 21-47.

This Page Intentionally Left Blank

Chapter 13
Non-Seismic Applications

Mary M. Poulton

1. INTRODUCTION

Neural networks have been applied to interpretation problems in well logging, and surface magnetic, gravity, electrical resistivity, and electromagnetic surveys. Since the geophysics industry is dominated by seismic acquisition and processing, the non-seismic applications of neural networks have not generated the same level of commercial interest. With the exception of well logging applications, most of the prolonged research into neural network applications for non-seismic geophysics has been government sponsored.

Although well logging and airborne surveys generate large amounts of data, most of the non-seismic techniques generate less data than a typical seismic survey. Minimal data processing is required for non-seismic data. After some basic corrections are applied to gravity and magnetic data, they are gridded and contoured and the interpreter works with the contoured data or performs some relatively simple forward or inverse modeling. Electrical resistivity data are plotted in pseudo-section for interpretation and also inverted typically to a 1D or 2D model. Electromagnetic data are often plotted in profile for each frequency collected (or gridded and contoured if enough data are collected) and also inverted to a 1D or 2D model. As desktop-computing power has increased, 3D inversions are being used more frequently. Some techniques such as electrical resistance tomography (ERT), a borehole-to-borehole imaging technique, collect large amounts of data and use rapid 3D inversions for commercial applications. The time-consuming part of an inversion is the forward model calculation. Neural network applications that produce estimates of earth-model parameters, such as layer thickness and conductivity, rely on forward models to generate training sets. Hence, generating training sets can be time consuming and the number of training models can be enormous. For applications where the training set size can be constrained, neural network "inversion" can be as accurate as least-squares inversion and significantly faster. Alternatively, neural networks can be trained to learn the forward model aspect of the problem and when coupled with least-squares inversion can result in orders of magnitude faster inversion.

As data acquisition times are decreased for the non-seismic techniques, the amount of data collected will increase and I believe we will see more opportunity for some specialized neural network interpretation. Surveys for unexploded ordnance (UXO) detection will undoubtedly exploit not only the rapid recognition capability of neural networks but also their ability to easily combine data from multiple sensors. Geophysical sensors attached to excavation tools ranging from drills to backhoes will provide feedback on rock and soil conditions and allow the operator to "see" ahead of the digface. The continuous data stream from these sensors will require a rapid processing and interpretation tool that provides the operator with an easily understood "picture" of the subsurface or provides feedback to the excavation equipment to optimize its performance. Real-time interpretation of data from geophysical sensors will probably emphasize classification of the data (both supervised and unsupervised). The first level of classification is novelty detection where a background or normal signature represents one class and the second class is the anomalous or "novel" signature. Metal detectors are essentially novelty detectors. The second level of classification is a further analysis of the novel signature. The final stage of interpretation may involve some estimation of the target parameters, such as depth of burial, size, and physical properties. All three interpretations can be performed simultaneously with data collection.

The chapters in this section of the book explain in detail issues related to training set construction, network design, and error analysis for airborne and surface frequency-domain electromagnetic data interpretation, surface time-domain electromagnetic data interpretation and galvanic well logs. In the remainder of this chapter, I review some of the other applications of neural networks for non-seismic geophysical data interpretation.

2. WELL LOGGING

The neural network applications in well logging using logs other than sonic have focused on porosity and permeability estimation, lithofacies identification, layer picking, and inversion. A layer picking application for unfocused galvanic logs is described in Chapter 15. Inversion applications for galvanic logs are described in Zhang et al. (1999). The focus of this section is on the porosity / permeability applications as well as the lithofacies mapping.

2.1. Porosity and Permeability estimation

One of the most important roles of well logging in reservoir characterization is to gather porosity and permeability data. Coring is both time consuming and expensive so establishing the relationship between petrophysical properties measured on the core in the laboratory and the well log data is vital. The papers summarized in this section use neural networks to establish the relationship between the laboratory-measured properties and the log measurements. The key to success in this application is the ability to extend the relationship from one well to another and, perhaps, from one field to another.

Good estimates of permeability in carbonate units are hard to obtain due to textural and chemical changes in the units. Wiener et al. (1991) used the back-propagation learning algorithm to train a network to estimate the formation permeability for carbonate units using LLD (laterolog deep) and LLS (laterolog shallow) log values, neutron porosity, interval transit time, bulk density, porosity, water saturation, and bulk volume water as input. Data were from the Texaco Stockyard Creek field in North Dakota. The payzone in this field is dolomitized shelf limestone and the porosity and permeability are largely a function of the size of the dolomite crystals in the formation. The relationship between porosity and permeability was unpredictable in this field because some high porosity zones had low permeability. The training set was created using core samples from one well. The testing set was comprised of data from core samples from a different well in the same field. The network was able to predict the permeabilities of the test samples with 90% accuracy, a significant improvement over multiple linear regression.

While not a porosity estimation application, Accarain and Desbrandes (1993) showed that an MLP trained with the extended delta bar delta algorithm could estimate formation pore pressure given porosity, percent clay, P-wave velocity, and S-wave velocity as input. Laboratory data from core samples were used for training. The cores were all water-saturated sandstone and were initially collected to test the effect of porosity and clay content on wave velocities. A total of 200 samples were used for training and another 175 for testing. Log data from four wells in South and West Texas were used for validation. The validation data from four wells produced an R^2 value equal to 0.95.

One approach to estimating porosity and permeability is to find a relationship between well log and laboratory data that includes all lithofacies within the reservoir. Such an approach is usually referred to as a non-genetic approach. The genetic approach is to find the relationship for each dominant lithofacies. Wong et al. (1995) use data already classified by lithofacies and then estimate the porosity and permeability values with separate networks. The porosity estimate from the first network was used as input to the permeability network. The lithofacies was coded with values from 1 to 11 for input to the network. Additional inputs were values from density and neutron logs and the product of the density and neutron values at each point in the log. Data from 10 wells in the Carnarvon Basin in Australia were used. A total of 1,303 data samples were available. Training data (507 samples) were extracted based on log values falling between the 25th and 75th percentiles for each lithofacies. The test set contained the remaining 796 patterns that were considered to deviate from the training data because of noise. A sensitivity analysis of the networks indicated that lithofacies information was by far the most important variable in predicting porosity and porosity plus density log were the most important variables in predicting permeability. Wireline log data produce smoother porosity predictions than core data because of the bulk sampling effect of the sonde. Hence, the porosity curves produced by the network were somewhat more difficult to interpret because of the lack of information from thin beds in the reservoir. To overcome this effect, the authors added "fine-scale" noise to the estimated porosity value based on the standard error for each lithofacies multiplied by a normal probability distribution function with zero mean and unit variance. For the human interpreter working with the results, the match to the core data was improved by adding noise to the estimate because it made the porosity values estimated from the log “look” more like the core data the interpreter was used to examining.

2.2. Lithofacies mapping

As we saw in the previous section, the determination of lithofacies is an important stage in subsequent steps of reservoir characterization, such as porosity and permeability estimation. Lithofacies mapping is usually a two step process involving segmenting a logging curve into classes with similar characteristics that might represent distinct lithofacies and then assigning a label to the classes, such as sandstone, shale, or limestone. Either supervised or unsupervised neural networks can be used to classify the logging data and then a supervised network can be used to map each class signature to a specific rock type. Baldwin et al. (1990) created some of the excitement for this application when they showed that a standard Facies

Analysis Log (FAL) took 1.5 person days compared to two hours to produce the same interpretation with a neural network, and that was for only one well.

In simple cases, it may be possible to skip the first step and map a logging tool response directly to a labeled class using a supervised network. McCormack (1991) used spontaneous potential (SP) and resistivity logs for a well to train a neural network to generate a lithology log. The lithologies are generalized into three types of sedimentary rocks: sandstone, shale, and limestone. He used a three layer neural network with two input PEs, three output PEs and five hidden PEs. One of the input nodes accepted input from the SP log and the other accepted data from the resistivity log for the same depths. The output used 1-of-n coding to represent the three possible lithologies. The result of the network processing is an interpreted lithology log that can be plotted adjacent to the raw log data.

A suite of logs can be used as input to the network rather than just SP and resistivity. Fung et al. (1997) used data from a bulk density log, neutron log, uninvaded zone resistivity, gamma ray, sonic travel time, and SP as input to a SOM network. The SOM clusters the log data into nine classes. The class number assigned to each pattern by the SOM network is appended to the input pattern and fed into an LVQ network which is a supervised classification based on a Kohonen architecture (see Chapter 5). The LVQ network maps the nine SOM classes into three user-defined classes of sandstone, limestone, and dolomite. The LVQ network performs the lithofacies identification needed for the genetic processing described by Wong et al. (1995) in the previous section. Data from each lithofacies can then be routed to a MLP network to estimate petrophysical properties such as porosity. The fit to core data of the MLP-derived estimates was better when the SOM and LVQ networks were used to classify the data compared to using only an MLP with back-propagation learning to perform all the steps in one network.

The identification of rock types from wireline log data can be more sophisticated than the major classes of clastics and carbonates. Cardon et al. (1991) used five genetic classes for a group of North Sea reservoirs that originated in a coastal plain environment during the Jurassic period: channel-fill; sheet-sand; mouthbar sand; coal; and shale. Geologists selected 13 features from wireline logs that they considered to be most important in discriminating between these genetic rock types. An interval in a well was selected for training and the input for the interval consisted of the interval thickness, average values and trends of the gamma ray log, formation density log, compensated neutron log, and borehole compensated sonic log. Also included were the positive and negative separations between the compensated neutron and formation density logs and between the gamma ray and borehole compensated sonic logs. The network was trained on 334 samples using an MLP with 5 hidden PEs and back-propagation learning. The network was tested on 137 samples. The network was correct in 92% of the identifications and where mistakes were recorded, the rock type was considered ambiguous by the geologists and not necessarily a mistake by the network. For comparison, linear discriminant analysis on the same data set yielded an accuracy of 82%.

The Ocean Drilling Program encountered a greater variety of lithologies than found in most reservoirs. Hence, a very robust method of automating lithofacies identification was highly desirable. Benaouda et al. (1999) developed a three-stage interpretation system that first statistically processed the log data, selected a reliable data set and finally performed the classification. When core recovery was poor and it was not known a priori how many different lithologies might be present, an unsupervised statistical classification was performed. Wireline data were reduced by a principal components analysis (PCA) and the PCA data clustered with a K-means algorithm. Intervals with core recovery greater than 90% were selected from the data set. The depth assignments of the core values were linearly stretched to cover 100% of the interval to match the well log data. The training class with the smallest population determined the size of all other training classes to avoid biasing the training by having class populations of very different sizes. An MLP using the extended delta bar delta learning algorithm was employed with an architecture of 15 input PEs, 15 hidden PEs, and 4 output PEs. ODP Hole 792E, drilled in the forearc sedimentary basin of the Izu-Bonin arc south of Japan was the data source for the study. The 250 m study interval contained five major depositional sequences. Sediments encountered in the hole were vitric sands and silts, pumiceous and scoriaceous gravels and conglomerates and siltstones. The PCA and K-means clustering of the well log data suggested that only four classes could be determined from the logs: volcanic-clast conglomerate; claystone-clast conglomerate; clay; siltstone. The neural network was consistently more accurate than the discriminant analysis. When all the data for a training class were included in the training set rather than restricting class size to the smallest class population, the accuracy improved as much as 7%. Biasing the training set was not a problem in this application. The best neural network had an accuracy of 85% compared to the best discriminant analysis accuracy of 84%. The discriminant analysis method, however, ranged from 55% to 85% in accuracy depending on the exact method employed. The results for both classifiers on intervals with poor core recovery was somewhat mixed although the network showed better agreement with the interpreters than the discriminant analysis classification.

Most neural network experiments use data from a small area within a field and a small number of wells. The same service company typically supplies the wireline data. Malki and Baldwin (1993) performed a unique experiment in which they trained a network using data from one service company's tools and tested the network using data from another company's tools. One hole containing 12 lithofacies was used for the study. The logs used in the study were short-spaced conductivity, natural gamma ray activity, bulk density, photoelectric effect, and neutron porosity. Schlumberger Well Services and Haliburton Logging services provided their versions of these tools. There were several differences between the two data sets: the Schlumberger tools were run first and the hole enlarged before the Haliburton tools were run; the two tools were designed and fabricated differently; some of the Schlumberger data was recorded at 0.5 ft increments and others at 0.1 ft increments while the Haliburton data was collected at 0.25 ft increments. A petrophysicist performed a visual interpretation on the data to create the training set. In trial 1 the network was trained on the Schlumberger data and tested on the Haliburton data and in trial 2 the sequence was reversed. They found better results when both data sets were normalized to their own ranges and the Haliburton data were used for training and the Schlumberger data were used for testing. The Haliburton data were better for training because the borehole enlargements produced "noise" in the data that could be compensated for by the network during training but not during testing. When the two data sets were combined, the best results were obtained. Lessons learned from this study were to include both "good" and "bad" training data to handle noisy test data, include low-resolution data in the training set if it might be encountered during testing, and test several network sizes.

While the previous studies were from the petroleum industry, there are applications for lithologic mapping in the mining industry as well. Huang and Wanstedt (1996) used an approach similar to other authors in this section to map well log data to classes of "waste rock", "semi-ore", and "ore". The geophysical logs included gamma ray, density, neutron, and resistivity. The logging data were compared to core logs and assays from the three boreholes measured in the experiment. Each tool was normalized to a range of (0,1). Twenty depth intervals for training in one borehole were selected and the average log values in the interval were input to an MLP network. The output used 1-of-n coding for the three classes. The network was tested on data from two other boreholes. Differences between the neural network classification and that based on the core analysis were negligible except for one 6-m interval. The core assay suggested waste for most of this interval but the network suggested ore or semi-ore. The interval contained disseminated metals that gave a sufficient geophysical response to suggest ore or semi-ore while the assay did not indicate a sufficient threshold for such a classification. As we have seen in previous examples, such discrepancies should not be viewed as blunders by the network so much as the normal geological ambiguity we always encounter.

3. GRAVITY AND MAGNETICS

Pearson et al. (1990) used high-resolution aeromagnetic data to classify anomalies as suprabasement or intrabasement in the Northern Denver-Julesberg Basin. Some Permo-Pennsylvanian reservoirs are trapped in structures on paleotopographic highs that are related to basement highs. The basement highs produce a subtle magnetic anomaly that can be spotted in profiles by an interpreter. Given the large amount of data collected in an aeromagnetic survey, a faster way to detect and classify these subtle features was desired. An MLP with back-propagation learning was given 10 inputs related to the magnetic data and various transforms, such as vertical and horizontal gradients. The network used two output PEs to classify signatures as suprabasement or intrabasement. The training set used both field data and synthetic models to provide a variety of anomalies. The network was then tested on more field data and more synthetic data. Anomalies identified by the network were compared to seismic and well log data for verification. The network located 80% of the structural anomalies in the field data and 95% of the structures in the synthetic data.

Guo et al. (1992) and Cartabia et al. (1994) present different ways of extracting lineament information from magnetic data. Guo et al. (1992) wanted to classify data into the eight compass trends (i.e. NS, NE, NW, etc.). A separate back-propagation network was created for each compass direction. The networks were trained with 7x7 pixel model windows. Field data were then input to the networks in moving 7x7 windows and the network with the largest output was considered the trend for that window.

Cartabia et al. (1994) used a Boltzmann Machine architecture, similar to the very fast simulated annealing method presented by Sen and Stoffa (1995), to provide continuity to pixels identified by an edge detection algorithm using gravity data. The edge detection algorithm does not provide the connectedness or thinness of the edge pixels that is required for a lineament to be mapped. By applying an optimization network, such as the Boltzmann

Machine to the edge pixels, a lineament map could be automatically produced that matched that produced by an expert interpreter.

Taylor and Vasco (1990) inverted gravity gradiometry data with a back-propagation learning algorithm. Synthetic models were generated of a high-density basement rock and a slightly lower density surficial deposit. The models were discretized into 18 cells and the network was required to estimate the depth to the interface at each cell. The average depth to the interface was 1.0 km. The training set was created by randomly selecting the depths to the interface and calculating the gravity gradient for the random model. The network was expected to estimate the depth given the gradient data. The network was tested on a new synthetic model that consisted of a north-south trending ridge superimposed on the horizontal basement at 10.0-km depth. The network was able to adequately reproduce the test model with only small errors in the depths at each cell location.

Salem et al. (2000) developed a fast and accurate neural network recognition system for the detection of buried steel drums with magnetic data. Readings from 21 stations each 1 m apart along a profile were used as input. The output consisted of two PEs that estimated the depth and horizontal distance along the profile for a buried object. To simulate the signature from a steel drum, forward model calculations were made, based on an equivalent dipole source. The drum was modeled at depths ranging from 2 m to 6 m at various locations along the profile. A total of 75 model responses were calculated for the training set. Noise was added to the data by simulating a magnetic moment located at the 10 m offset of the profile line at a depth of 2.1 m. Noise ranging from 10% to 40% was added to the data. The network estimates of the drum location were acceptable with up to 20% noise. Data from 10 profiles at the EG&G Geometrics Stanford University test site were used to test the network. On average, the depths of the barrels were estimated with 0.5 m. The offset location estimates were less accurate but in most cases were within one barrel dimension of the true location (barrels were 0.59 m diameter and 0.98 m height).

4. ELECTROMAGNETICS

4.1. Frequency-Domain

Cisar et al. (1993) developed a neural network interpretation system to locate underground storage tanks using a Geonics EM31-DL frequency-domain electromagnetic instrument. The sensor was located on a non-conductive gantry and steel culverts were moved under the sensor while measurements were recorded. Three different vertical distances between the sensor and target were used. The orientation of the target relative to the sensor was also varied. Data were collected as in-phase and quadrature in both the horizontal and vertical dipole modes. The input pattern vector consisted of the four measurements recorded at three survey locations approximately 2 m apart plus the ratio of the quadrature to in-phase measurements for both dipole configurations. Hence the input pattern contained 18 elements. Three depths of burial for the target were considered 1.2 m, 2.0 m, and 2.4 m. For each depth of burial, two output PEs are coded for whether the target is parallel or perpendicular to the instrument axis. Hence the network is coded with 6 output PEs. When tested with data collected at Hickam Air Force Base in Hawaii, the neural network produced a location map of buried underground storage tanks that matched that produced by a trained interpreter.

Poulton et al. (1992a, b), Poulton and Birken (1998), Birken and Poulton (1999), and Birken et al. (1999) used neural networks to interpret frequency-domain electromagnetic ellipticity data. Poulton et al. (1992a,b) focused on estimating 2D target parameters of location, depth, and conductivity of metallic targets buried in a layered earth. A suite of 11 frequencies between 30 Hz and 30 kHz were measured at each station along a survey line perpendicular to a line-source transmitter. The data were gridded to form a 2D pseudosection. Efforts were made to study the impact of the data representation and network architecture on the overall accuracy of the network's estimates. In general, smaller input patterns produced better results, provided the smaller pattern did not sacrifice information. The entire 2D image contained 660 pixels. A subsampled image contained 176 pixels. The major features of the data, the peak and trough amplitudes and locations for each frequency along the survey line (see Figure 4.5 for an example of an ellipticity profile) produced an input pattern with 90 PEs. Using the peak alone required 30 input PEs (peak amplitude and station location for each of 15 gridded frequencies). A 2D fast Fourier Transform required four input PEs. The Fourier transform representation produced results that were comparable to using the entire image as an input pattern.

Several learning algorithms were tested as well: directed random search, extended delta bar delta, functional link, back-propagation, and self-organizing map coupled with back-propagation. The directed random search and functional link networks did not scale well to large input patterns but performed very accurately on small input patterns. The hybrid network of the self-organizing map, coupled with back-propagation proved the most versatile and overall most accurate network for this application.

Poulton and Birken (1998) found that the modular neural network architecture (described in more detail in Chapter 15) provided the most accurate results for 1D earth model parameter estimation, using ellipticity data in a frequency range of 1 kHz to 1 MHz. The 11 recorded ellipticity values did not contain enough information for interpretations beyond three earth layers; so, the training set was constrained to two and three layers. Three different transmitter-receiver separations were typically used in the field system and a different network was required for each. For each transmitter-receiver separation, training models were further segregated according to whether the first layer was conductive or resistive. Hence, the interpretation system required 12 separate networks. Since each network takes only a fraction of a second to complete an interpretation, all 12 were run simultaneously on each frequency sounding. A forward model was calculated based on each estimate of the layer thickness and resistivities. The forward model calculations were compared to the measured field data and the best fit was selected as the best interpretation. Error analysis of the network results was subdivided based on resistivity contrast of the layers and thickness of the layers. Such analysis is based on the resolution of the measurement system and not the network's capabilities. There was no correlation found between accuracy of the resistivity estimates and the contrast of the models. Estimates of layer thickness were dependent on layer contrast. Estimates of layer thickness less than 2 m thick for contrasts less than 2:1 were unreliable. The modular network was examined to see how it subdivided the training set. Each of the five expert networks responded to different characteristics of the ellipticity sounding curves. One expert collected only models with low resistivities. The second expert grouped models with first-layer resistivities greater than 200-ohm meters. The third expert selected models with high contrast and thick layers. The fourth expert picked models with low contrast and thick layers. The fifth expert responded to all the remaining models.

Birken and Poulton (1999) used ellipticity data in a frequency range 32 kHz to 32 MHz to locate buried 3D targets. In the first stage of interpretation, radial basis function networks were used to create 2D pseudosections along a survey line. The pseudosections were based on 1D interpretations of pairs of ellipticity values at adjacent frequencies. While the actual model parameters produced by the 1D interpretation over a 3D target are inaccurate, a consistent pattern was observed in the 2D pseudosections that reliably indicated the presence of a 3D body. Hence, the technique could be used to isolate areas that require the more computationally intensive 3D inversion. Another network was used to classify individual sounding curves as being either target or background. Data from targets buried at the Avra Valley Geophysical Test Site near Tucson, Arizona were used as the primary training set. The test set consisted of data from a waste pit at the Idaho National Engineering and Environmental Laboratory (INEEL) near Idaho Falls, Idaho. The test results were poor when only the Avra Valley data were used for training. When four lines of data from INEEL were included, the test results achieved 100% accuracy. The authors concluded that data sets from different field sites can be combined to build a more robust training set. Training times for a neural network are short enough that networks can be retrained on site as new data are acquired.

4.2. Time-Domain

Gifford and Foley (1996) used a neural network to classify signals from a time-domain EM instrument (Geonics EM61) for a UXO (unexploded ordnance) application. One network classified the data as being from UXO targets larger or smaller than 2 pounds. The second network estimated the depth to the target. The success of this application was a result of a comprehensive training set and pre-processing the data. The authors constructed an extensive knowledge base of field data from UXO surveys around the country. The database contained geophysical data, GIS coordinates and the type of object that generated the response as well as the depth of burial of the object. The database contained data from both UXO and non-UXO targets. Data acquired with the EM61 instrument were normalized to a neutral site condition. The resulting input pattern contained 15 elements from each sample point in a survey. Two channels of data were collected with the EM61. Many of the 15 input elements described relationships between the two channels and include differences, ratios, and transforms of the channels. An MLP trained with conjugate gradient and simulated annealing was used for the application. After training on 107 examples of UXO signatures, the network was tested on an additional 39 samples. Analysis of the results indicated that 87% of the samples were correctly classified as being heavier or lighter than 2 lbs. Of the targets lighter than 2 pounds, 90% were correctly identified. Of the targets heavier than 2 pounds, 7 out of 9 samples were correctly classified. The authors calculated a project cost saving of 74% over the conventional UXO detection and excavation methods with the neural network approach.

4.3. Magnetotelluric

Magnetotelluric data inversion was studied by Hidalgo et al. (1994). A radial basis function network was used to output a resistivity profile with depth given apparent resistivity values at 16 time samples. The output assumed 16 fixed depths ranging from 10.0 to 4,000 m. A cascade correlation approach to building the network was used (see Chapter 3 for description). The authors found that best results were obtained when the four general type curves were segregated into four different training sets (A=monotonic ascending, Q=monotonic descending, K=positive then negative slope, H=negative then positive slope). A post-processing step was added to the network to improve the results. The resistivity section output by the network was used to generate a forward model to compare to the field data. The RMS error between the network-generated data and the observed data was calculated. If the RMS error exceeded a user-specified threshold, the error functional was calculated as

$$U = \lambda \sum_{i} (s_i' - s'_{i+1})^2 k_i + \sum_{i} (e_i - \rho_i(s'))^2, \tag{13.1}$$

where s is the resistivity profile consisting of resistivity at 16 depths, k is set to 0 at discontinuities and 1 elsewhere, e is the network estimate of the resistivity, ρ(s) is the desired resistivity value. Hence, the first part of the equation is the model roughness and the second part is the least-squares error of the network estimate. The Jacobian matrix calculates the gradient of the error functional,

$$\frac{d\rho(s')}{d(s')}. \tag{13.2}$$

The output of the Jacobian matrix is used as input to a QuickProp algorithm that outputs a new resistivity profile. The authors show one example where a profile with an RMS error = 0.53 was moved to a new model with an RMS error = 0.09 by this method.

Few researchers have tackled 3D interpretations of electromagnetic data. Spichak and Popova (1998) describe the difficulties with modeling and inverting 3D electromagnetic data as related to incorporating a priori constraints, especially in the presence of noise and the large computational resources required for each interpretation. In monitoring situations where data need to be continuously interpreted, a new approach is required that can map recorded data to a set of geoelectric parameters. The key to this approach is making the neural network site or application specific to avoid the inherent parameterization problems involved in creating a training set that describes all possible earth models. Spichak and Popova (1998) created a training set for a 3D fault model, where the fault is contained in the second layer of a two-layer half-space model. The model was described by six parameters: depth to upper boundary of the fault (D), first layer thickness (H1), conductivity of first layer (C1), conductivity of second layer (C2), conductivity of the fault (C), width of fault (W), strike length of fault (L), and inclination angle of fault (A). Electric and magnetic fields were calculated for the models using audiomagnetotelluric periods from 0.000333 to 0.1 seconds. A total of 1,008 models were calculated. A 2D Fourier transform was applied to the calculated electromagnetic fields. The Fourier coefficients for five frequencies were used as the input to the network that in turn estimated the six model parameters. The authors performed a sensitivity analysis on the results to determine the best input parameters to use. The lowest errors were recorded when apparent resistivity and impedance phases at each grid location were used as input to the Fourier transform. The authors also performed a detailed analysis of the effect of noise on the training and test results. The authors conclude that neural networks can perform well on noisy data provided the noise level in the training data matches that of the test data. When the training data have a much lower noise level than the test data, the accuracy of the estimated parameters is greatly diminished.

4.4. Ground Penetrating Radar

Ground penetrating radar (GPR) is a widely used technique for environmental surveys and utility location. The processing techniques used for GPR data are similar to those used for seismic data. However, none of computational neural network processing schemes described in Part II of this book have been applied to GPR data.

Two papers have been found in the literature on neural networks applied to GPR data. Poulton and El-Fouly (1991) investigated the use of neural networks to recognize hyperbolic reflection signatures from pipes. A logic filter and a set of cascading networks were used as a decision tree to determine when a signature came from a pipe and then determine the pipe composition, depth, and diameter.

Minior and Smith (1993) used a neural network to predict pavement thickness, amount of moisture in surface layer of pavement, amount of moisture in base layer, voids beneath slabs, and overlay delamination using ground penetrating radar data. For practical application, the GPR system needed to be towed at highway speeds of 50 mph with continuous monitoring of the received GPR signal. Such a large data stream required an automated interpretation method. A separate back-propagation network was trained for each desired output variable. Synthetic models were used for training because of the wide range of pavement conditions that could be simulated. The input pattern consisted of a sampled GPR wave with 129 values. All of the data prior to the second zero crossing of the radar trace were discarded. The trace was then sampled at every second point until 128 values had been written. The authors found that adding noise to the training data was crucial for the network to learn the most important features of the radar signal. The neural networks located voids within 0.1 inch; moisture content was estimated within 0.1% and the network could reliably distinguish between air and water filled voids.

5. RESISTIVITY

Calderon-Macias et al. (2000) describe a very fast simulated annealing (VFSA) neural network used for inverting electrical resistivity data. The training data were generated from a forward model and the test data were taken from the published literature. A Schlumberger sounding method was used for the electrode configuration. Two hundred and fifty sounding curves were generated for three layer earth models where $\rho1>\rho2<\rho3$. The resistivity of the top layer was fixed at 1 Ω m for all models. The thickness of the first layer varied between 1.0 and 10.0 m. The resistivity of the second layer varied between 0.03 and 0.20 Ω m and the resistivity of the third layer was between 0.15 and 0.61 Ω m. The second layer thickness ranged between 3 m and 20 m. Twenty different electrode spacings were modeled. A hidden layer with 8 PEs was found to be optimal. The network-estimated model was used as a starting model for a least-squares inversion with Newton's method. While the neural network estimate based on field test data was close to the "true" model, the least-squares inversion improved the accuracy of the second layer thickness estimate. When a random starting model, within the boundaries of the training parameters was used for the least-squares inversion, the method did not converge to the correct model.

6. MULTI-SENSOR DATA

Brown and Poulton (1996) combined data from a frequency-domain electromagnetic sensor, Geonics EM38 Ground Conductivity Meter, and a GRS-1 Fluxgate Magnetometer to distinguish buried objects from the background soil response to then classify the objects as conductive or nonconductive, and finally to estimate their depth of burial. The data were collected as part of the Dig-face characterization experiment phase of the Buried Waste Integrated Demonstration Program at the Idaho National Engineering and Environmental Laboratory (INEEL). The goal of the experiment was to demonstrate that a high-resolution image of a buried waste site could be used to aid excavation by successively scanning an area, interpreting the data and excavating a thin layer of soil. Since the area to be excavated was scanned by the geophysical sensors with very small station spacing, large amounts of data were collected and required a rapid interpretation technique.

Data used for the experiment were collected over a mock-up of a hazardous waste dump called the Cold Test Pit (CTP). Some of the objects were actually simulated plumes of alcohol or saltwater and were used to test chemical rather than geophysical sensors. The objects were buried at varying depths from approximately 0.5 meters to 2.0 meters. The object types ranged from filing cabinets to wooden boxes with varying contents (metals, paper), iron and PVC pipes, 55-gallon drums, and small buckets. The sensors were mounted on a trolley that first scanned a line across the waste pit with 7 cm between readings. The trolley then moved to the next line, 15 cm from the previous, and scanned. After the entire pit was surveyed the sensors were lowered 15 cm to 30 cm and the pit was rescanned. When the sensors reached the soil surface a thin layer of soil was removed and the survey repeated.

The EM38 recorded in-phase and quadrature data for horizontal and vertical dipoles and the magnetometer measured total field and vertical gradient. Several neural networks were created to classify the data as representing target or background; the depth of the identified targets was estimated; and, finally, the targets were further classified as being conductive or non-conductive. The initial interpretations were performed on individual data points so that a picture of the subsurface could develop as data were acquired. The second set of networks used five adjacent data points along a survey line to perform the same interpretations. Finally, data from multiple scan levels were concatenated and another set of networks produced classifications of target versus background, depth estimates, and conductive versus non-conductive properties. Thus, networks could begin interpreting the data as soon as it was acquired. As more and more data were collected, the input pattern included more information and produced a more accurate picture of the subsurface.

The authors never had the opportunity to integrate their interpretation system with the data acquisition system so the purported advantages of this approach could not be fully explored.

REFERENCES

Accarain, P., and Desbrandes, R., 1993, Neuro-computing helps pore pressure determination: Petroleum Engineer International, Feb, 39-42.

Baldwin, J., Bateman, R., and Wheatley, C., 1990, Application of a neural network to the problem of mineral identification in well logs: The Log Analyst, September-October, 279-293.

Benouda, D., Wadge, G., Whitmarsh, R., Rothwell, R., and McLeod, C., 1999, Inferring lithology of borehole rocks by applying neural network classifiers to downhole logs: an example from the Ocean Drilling Program: Geophysics Journal International, **136**, 477-491.

Birken, R., Poulton, M., and Lee, K., 1999, Neural network interpretation of high frequency electromagnetic ellipticity data, Part I: Understanding the half-space and layered earth response: Journal of Environmental and Engineering Geophysics, **4**, 93-103.

Birken, R., and Poulton, M., 1999, Neural network interpretation of high frequency electromagnetic ellipticity data, Part II: Analyzing 3D responses: Journal of Environmental and Engineering Geophysics, **4**, 149-165.

Birken, R., and Poulton, M., 1997, Neural network interpretation of high-frequency electromagnetic ellipticity data: Proceedings of the SAGEEP '97, **1**, 381-390.

Brown, M., and Poulton, M., 1996, Locating buried objects for environmental site investigations using neural networks: Journal of Environmental and Engineering Geophysics, **1**, 179-188.

Calderon-Macias, C., Sen, M., and Stoffa, P., 2000, Artificial neural networks for parameter estimation in geophysics: Geophysical Prospecting, **48**, 21-47.

Cardon, H., Hoogstraten, R., and Davies, P., 1991, A neural network application in geology: Identification of genetic facies, *in* Kohonen, T., Makisara, K., Simula, O., and Kangas, J., Eds., Artificial Neural Networks, Elsevier Science Publishers, 809-813.

Cartabia, G., Zerilli, A., and Apolloni, B., 1994, Lineaments recognition for potential fields images using a learning algorithm for Boltzmann machines: Society of Exploration Geophysicists, 64th Annual International Meeting and Exposition, 432-435.

Cisar, D., Dickerson, J., and Novotny, T., 1993, Electromagnetic data evaluation using neural networks: Initial investigation - underground storage tanks: Proceedings of the SAGEEP '93, **2**, 599-612.

Fung, C., Wong, K., and Eren, H., 1997, Modular artificial neural network for prediction of petrophysical properties from well log data: IEEE Transactions on Instrumentation and Measurement, **46**, 1295-1299.

Gifford, M., and Foley, J., 1996, Neural network classification techniques for UXO applications: Proceedings of the SAGEEP '96, **1**, 701-710.

Guo, Y., Hansen, R., and Harthill, N., 1992, Feature recognition from potential fields using neural networks: Society of Exploration Geophysicists, 62nd Annual International Meeting and Exposition, 1-5.

Hidalgo, H., Gomez-Trevino, E., and Swiniarski, R., 1994, Neural network approximation of an inverse functional: IEEE World Congress on Neural Networks and Fuzzy Systems, Orlando, FL.

Huang, Y., and Wanstedt, S., 1996, Application of neural network model for ore boundary delineation based on geophysical logging data: IEEE International Conference on Neural Networks: IEEE, **4**, 2148-2153.

Malki, H., and Baldwin, J., 1993, On the comparison results of the neural networks trained using well-logs from one service company and tested on another service company's data: IEEE International Conference on Neural Networks: IEEE, **3**, 1776-1779.

McCormack, M., 1991, Neural computing in geophysics: The Leading Edge of Exploration, 11-15.

Minior, D., and Smith, S., 1993, Neural networks for highway maintenance investigations using ground penetrating radar: Proceedings of the SAGEEP '93, **1**, 449-462.

Pearson, W., Wiener, J., and Moll, R., 1990, Aeromagnetic structural interpretation using neural networks: A case study from the Northern Denver-Julesberg Basin: Society of Exploration Geophysicists, 60th Annual International Meeting and Exposition, 587-590.

Poulton, M., and Birken, R., 1997, Estimating one-dimensional models from frequency domain electromagnetic data using modular neural networks: IEEE Transactions on Geoscience and Remote Sensing, **36**, 547-555.

Poulton, M., Sternberg, B., and Glass, C., 1992a, Neural network pattern recognition of subsurface EM images: Journal of Applied Geophysics, **29**, 21-36.

Poulton, M., Sternberg, B., and Glass, C., 1992b, Location of subsurface targets in geophysical data using neural networks: Geophysics, **57**, 1534-1544.

Poulton, M., and El-Fouly, A., 1991, Pre-processing GPR signatures for cascading neural network classification: Society of Exploration Geophysicists, 61st Annual International Meeting and Exposition, 507-509.

Salem, A., Ushijima, K., Ravat, D., and Johnson, R., 2000, Detection of buried steel drums from magnetic anomaly data using neural networks: Proceedings of the SAGEEP 2000, **1**, 443-452.

Sen, M., and Stoffa, P., 1995, Global Optimization Methods in Geophysical Inversion: Elsevier.

Spichak, V., and Popova, I., 1998, Application of the neural network approach to the reconstruction of a three-dimensional geoelectric structure: Izvestia, Physics of the Solid Earth, **34**, 33-39, Translated from Fizika Zemli, 1998, 39-45.

Swiniarski, R., Hidalgo, H., and Gomez-Trevino, E., 1993, Neural networks applied in the geophysical inversion problem: Proc. SPIE Ground Sensing, **1941**, 151-158.

Taylor, C., and Vasco, D., 1990, Inversion of gravity gradiometry data using neural networks: Society of Exploration Geophysicists, 60th Annual International Meeting and Exposition, 591-593.

Wiener, J., Rogers, J., Rogers, J., and Moll, R., 1991, Predicting carbonate permeabilities from wireline logs using back-propagation neural networks: Society of Exploration Geophysicists, 61st Annual International Meeting and Exposition, 285-288.

Wong, P., Taggert, I., Gedeon, T., 1995, Use of neural network methods to predict porosity and permeability of a petroleum reservoir: AI Applications, **9**, 27-37.

This Page Intentionally Left Blank

Chapter 14
Detection of AEM Anomalies Corresponding to Dike Structures

Andreas Ahl and Wolfgang Seiberl

1. INTRODUCTION

For the past 10 years the Geological Survey of Austria (GSA), in cooperation with the Institute of Meterorology and Geophysics of the University of Vienna (IMG), has been performing airborne electromagnetic (AEM) measurements with a helicopter towed AEM system*. The advantages of airborne geophysical measurements over terrestrial surveys lie in the high quality of the data and in the significantly higher productivity. In addition, by using a helicopter, measurements can always be made under extreme topographical conditions (such as pathless alpine terrain).

The major geoscientific applications of AEM measurements are in the following areas:

- Investigation of groundwater resources
- Geotechnical applications (e.g. landslides, mass movements)
- Exploration of raw materials (mass raw materials like clay and gravel, ore resources)
- Assisting terrain mapping (geology)

Due to the enormous quantity of data which will be obtained during such measurements (about 10^5 measuring points for each survey area), most of the users of such systems use only simple mathematical-physical models, e.g. the homogeneous half-space (Fraser, 1978), the Schwerpunktstiefe (crucial point depth) (Sengpiel, 1988), etc. Chiefly these methods are used to keep the calculation time low.

At the IMG and the GSA the measured data are interpreted as a homogeneous half-space model or a two-layer half-space model (see Figure.14.1). In practice, based upon the measured values for each sampling point, the decision is first made whether the data should be be interpreted as a homogeneous half-space or as a two-layer half-space. After the classification for each measuring point the calculation of the parameters of the appropriate model takes place. Due to the high calculation speed, computational neural networks (CNN) are used to perform the classification and the calculation of the model parameters. As a final step the models for each measuring point are put together to form a three-dimensional (3D) model of the electrical conductance of the survey area.

---

* This work was supported by the Austrian Science Fund FWF (FWF-Grant No. P11833-GEO)

After making a pointwise interpretation based on the layered earth models, the next step in the analysis of AEM data is to interpret profile segments using 2D structures. As a first step we want to detect certain 2D structures based on AEM profile segments. In making a detection, we determine whether the 2D structure we are looking for is located in the middle of the observed profile segment. Such a detection of 2D structures has two decisive advantages: If this detection is done by a CNN, it is possible to quickly find the AEM anomalies of certain 2D structures (e.g. dikes). We can also mark off areas inside the survey area where layered earth models (1D models) are not suitable as a description for the subsurface. Because of its relevance in mineral exploration, we decided to use a conductive dike in a resistive environment as the 2D structure to be detected. Since CNNs are very fast in application, we decided to use CNNs for the automatic detection of 2D structures from the AEM data.

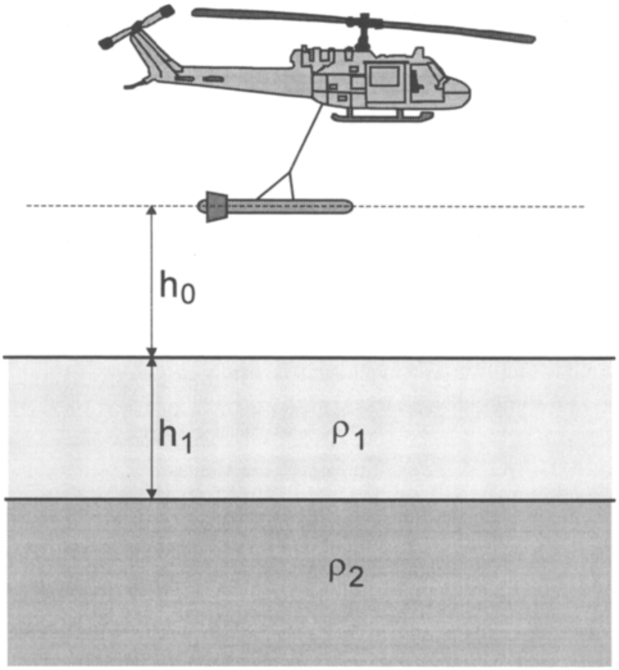


Figure 14.1. Airborne electromagnetic system over a 2 layer half-space.

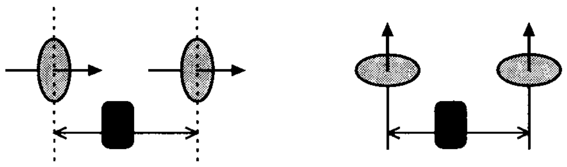
2. AIRBORNE ELECTROMAGNETIC METHOD - THEORETICAL BACKGROUND

2.1. General

Figure 14.2 shows a schematic drawing of the current airborne electromagnetic system used in Austria. In this helicopter towed AEM system, there are two horizontal coplanar and two vertical coaxial maximum coupled coil systems (see Table 14.1). The sensor is carried at a height h_0 above ground level.

Table 14.1
Parameters of the aeroelectromagnetic system

	Frequency	Coil separation	Configuration
f_1	434 Hz	4.53 m	horizontal coplanar
f_2	3212 Hz	4.53 m	vertical coaxial
f_3	7002 Hz	4.49 m	horizontal coplanar
f_4	34133 Hz	4.66 m	vertical coaxial



vertical coaxial and horizontal coplanar configuration

Figure 14.2. Coil orientations used in the AEM system.

A voltage with a certain frequency (f_1, f_2, f_3, f_4) is applied to each coil system, which consists of a transmitter and receiver coil with separation r. When the sensor system is located in free space (which is about 500 m above ground level) an electromagnetic field (primary field) is generated which induces a voltage U_P in the receiver coil. In this case the primary field is the only cause for the voltage induced in the receiver coil. If the sensor system is near the surface, the primary field stimulates a secondary field in the subsurface. This secondary field induces an additional voltage U_S in the receiver coil. To measure only the secondary field, the influence of the primary field is compensated by a compensation coil (bucking coil).

As there is a phase shift between the primary and the secondary field, it has been proven to be convenient to name the component of the measured secondary field which has a phase shift of 180° to the primary field the real or inphase component. The component of the measured secondary field, which has a phase shift of 90° to the primary field, is called the imaginary, out-of-phase or quadrature component. The measured components of the complex dimensionless ratio U_S/U_P are given in ppm (parts per million).

2.2. Forward modeling for 1 dimensional models

A special case, for which it is possible to describe the dependance of the measured values on the parameters of the assumed model, is a horizontally layered half-space with N layers. In figure 14.1, the subsurface is described as a 2 layer half-space model.

From electromagnetic theory (Wait, 1982) one can derive formulas for U_S/U_P as a function of the frequency f, the coil separation r, the sensor height h_0, the earth layer resistivities ρ_1 and ρ_2, the air resistivity ρ_0 and, the thickness h_1 of the first layer. μ_0 and ε_0 denote the magnetic permeability and the dielectric permittivity of free space:

horizontal coplanar :

$$\left(\frac{U_S}{U_P}\right)(r, \omega, \rho_1, \rho_2, h_0, h_1) = 1 + B^3\, T_0 . \tag{14.1}$$

vertical coaxial :

$$\left(\frac{U_S}{U_P}\right)(r, \omega, \rho_1, \rho_2, h_0, h_1) = -2\left(1 + \frac{B^2}{2}(T_2 - B\, T_0)\right). \tag{14.2}$$

B = r/δ , where δ is the skindepth of the first layer :

$$\delta = \sqrt{\rho_1/\pi f \mu_0} . \tag{14.3}$$

T_0 and T_2 are integrals defined by:

$$\begin{aligned} T_0 &= -\delta^3 \int_0^{+\infty} \lambda^2\, R_{TE}(\lambda)\, e^{-2\lambda h_0}\, J_0(\lambda r)\, d\lambda \\ T_2 &= -\delta^2 \int_0^{+\infty} \lambda\, R_{TE}(\lambda)\, e^{-2\lambda h_0}\, J_1(\lambda r)\, d\lambda \end{aligned} \tag{14.4}$$

where $J_0(x)$ and $J_1(x)$ are Besselfunctions of 0. and 1. order.

For the 2 layer models the coefficient R_{TE} appearing in the integrals T_0 and T_2 are defined by

$$R_{TE} = \frac{\alpha_0 - \alpha_1 \dfrac{\alpha_2 + \alpha_1 \tanh(\alpha_1 h_1)}{\alpha_1 + \alpha_2 \tanh(\alpha_1 h_1)}}{\alpha_0 + \alpha_1 \dfrac{\alpha_2 + \alpha_1 \tanh(\alpha_1 h_1)}{\alpha_1 + \alpha_2 \tanh(\alpha_1 h_1)}} . \tag{14.5}$$

with $\alpha_i = \sqrt{\lambda^2 + j\,\omega\mu_0\,(1/\rho_i + j\,\omega\varepsilon_0)}$ $\quad i = 0, 1, 2\ ;\ j = \sqrt{-1}$ and $\omega = 2\pi f$

and in the case of homogeneous half-space models ($\rho_1 = \rho_2$) :

$$R_{TE} = \frac{\alpha_0 - \alpha_1}{\alpha_0 + \alpha_1}. \tag{14.6}$$

The computation of the equations (14.1) and (14.2) for any number of layers is implemented by a computer program. From the structures of equations (14.1) and (14.2), we see that the calculation of ρ_1, ρ_2 und h_1, based on the measured values of U_S/U_P for different frequencies and coil configurations, can only be done by making costly numerical approximations.

2.3. Forward modelling for 2 dimensional models with EMIGMA

In the following case, the forward modelling of the 2D models will be carried out with the aid of EMIGMA®, a software package that calculates the EM anomalies of 3D structures (Petros Eikon, Inc., 1997).

There are three numerical procedures that may be used to calculate the forward modelling of 2D or 3D structures :

1. Finite Differences Method (FDM)
2. Finite Elements Method (FEM)
3. Integral Equations Method (IE)

All three methods are based on the 'method of weighted residuals' (Harrington, 1968). They are used to solve an equation of the form L(f) = g. *L* may be a differential operator (as in procedures 1 and 2) or an integral operator (as in procedure 3); *f* is an unknown vector or a scalar field, and *g* is a known term that describes the source.

In the Differential Equations Method (procedures 1 and 2) for a 3D body, the total electrical field has to be calculated over a grid in the body and the medium. However with the Integral Equations Method (procedure 3,) you are only required to calculate the unknown electrical field in the body. EMIGMA is based on the Integral Equations Method (Hohmann, 1975).

In the Integral Equations Method a simple background medium structure will be assumed (homogeneous or layered half-space) so that the field in this material can be calculated analytically or quasianalytically.

The calculation of the EM field within the body is done in three steps :

- The first step is to calculate the electrical field at the body produced by the source in the presence of the surrounding medium.
- Secondly, the scattering currents, which are produced by the occuring field, are to be calculated. To calculate these currents the body is first divided into cells which are replaced by an equivalent current distribution. To determine the current distribution in the body, the program EMIGMA uses the so-called 'LN scattering algorithm' (Petros Eikon, Inc., 1997).

- Finally, the secondary EM field, which is created by the scattering currents, can be calculated at every point outside of the body.

While using the 'LN scattering algorithm', one observes that exact results are produced only in the case of current channelling.

3. FEEDFORWARD COMPUTATIONAL NEURAL NETWORKS (CNN)

For all CNNs we used a feedforward multilayer perceptron architecture. In the standard architecture, connections only exist between units in adjacent layers. As an extension, we also use connections from one layer to all following layers.

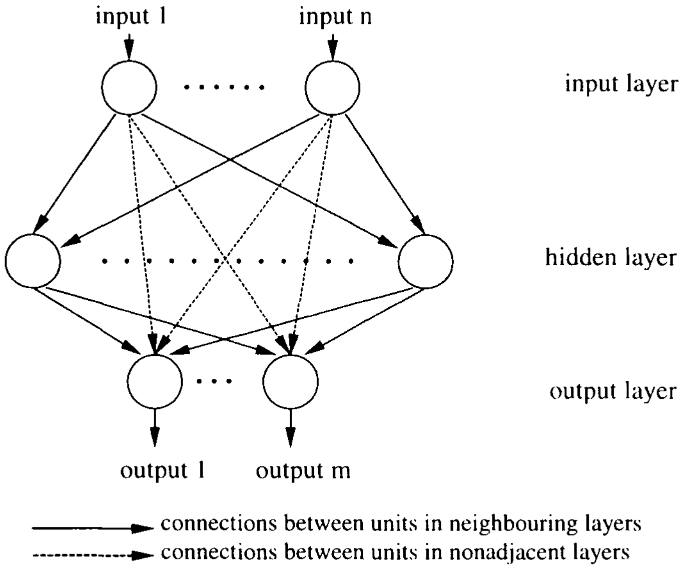
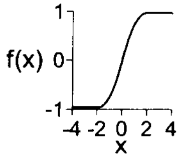
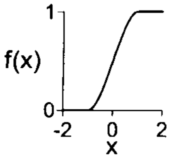
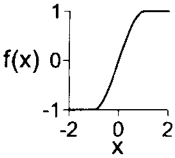
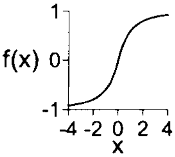
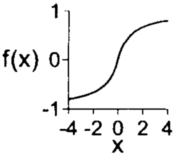

input 1
input n
input layer
hidden layer
output layer
output 1
output m
connections between units in neighbouring layers
connections between units in nonadjacent layers

Figure 14.3. Architecture of a feedforward CNN.

Adding direct connections from the input layer to the output layer (Figure 14.3) can speed up training. Especially when the function to be approximated is almost linear and it needs only a small amount of adjustment from nonlinear hidden layer units. This method can also cut down on the required number of hidden layer units. The virtue of the direct connection is discussed by Sontag (1990).

Implementing transfer function using exponentials can be computer-intensive. To save on CPU time, we used numerically simpler implementations of sigmoid transfer functions (see Table 14.2).

Table 14.2
Transfer functions

Name	x	f(x)	Graph
hyperbolic_tangent_11 (Anguita et al., 1993)	$x \leq -1.92033$	-0.96016	
	$-1.92033 < x < 0$	$-0.96016 + 0.26037 \cdot (x + 1.92033)^2$	
	$0 \leq x < 1.92033$	$0.96016 - 0.26037 \cdot (x - 1.92033)^2$	
	$x \geq 1.92033$	0.96016	
cubic_sigmoid_01	$x < -1$	0	
	$-1 \leq x \leq +1$	$0.5 + 0.75 \cdot x - 0.25\, x^3$	
	$x > +1$	1	
cubic_sigmoid_11	$x < -1$	-1	
	$-1 \leq x \leq +1$	$1.5 \cdot x - 0.5 \cdot x^3$	
	$x > +1$	1	
Hypsigmoid_11	$x \leq 0$	$-1 + 2/((1-x)^2 + 1)$	
	$x > 0$	$1 - 2/((1+x)^2 + 1)$	
elliot_sigmoid_11 (Elliott, 1993)	$x < -1000$	-1	
	$-1000 \leq x \leq 0$	$x/(1-x)$	
	$0 < x \leq 1000$	$x/(1+x)$	
	$x > 1000$	1	

For the output units we always used transfer functions of cubic_sigmoid type. In the hidden layers, we not only tried combinations of the transfer functions listed in Table 14.2, but also hypsigmoid and hyperbolic tangent transfer functions with output values running from 0 to 1. Best generalization results were achieved with networks using asymmetrical transfer functions (output values centered around 0) in the hidden layers.

The networks were trained with a pattern mode back-propagation learning algorithm, including a momentum term, to adjust the weights $\bar{w}$ (learn rate η_w, momentum α_w) and the gains $\Delta\beta$(learn rate η_β, momentum α_β) of the CNN according to equations (14.7) to (14.10).

$$\Delta\bar{w}(t) = -\eta_w \frac{\partial \bar{E}^\mu}{\partial \bar{w}(t)} + \alpha_w \cdot \Delta\bar{w}(t-1) \,. \tag{14.7}$$

$$\Delta\vec{\beta}(t) = -\eta_\beta \frac{\partial \bar{E}^\mu}{\partial \vec{\beta}(t)} + \alpha_\beta \cdot \Delta\vec{\beta}(t-1). \tag{14.8}$$

$$\vec{w}(t+1) = \vec{w}(t) + \Delta\vec{w}(t). \tag{14.9}$$

$$\vec{\beta}(t+1) = \vec{\beta}(t) + \Delta\vec{\beta}(t) \ . \tag{14.10}$$

In pattern mode, the updating of weights and gains according to the CNN output error E^μ at the μ-th training sample is done directly after the representation of this training sample.

Because the standard gradient descent approach of the back-propagation learning algorithm is slow to converge, some modifications were used to speed up training. When the output of a neuron is near its upper or lower bound, the derivative of the sigmoid transfer function approaches zero (saturated neuron). Therefore, the changes for weights connected to a saturated neuron will also approach zero. This can slow down the learning process significantly. Adding a small offset (e.g. 0.1) to the derivative of the transfer functions of all units counteracts this problem (Fahlman, 1988).

Another method to accelerate training is to calculate the error signals of the output units as the hyperbolic arctangent (Fahlman, 1988) of the unit's output errors, taking on values between −1 and +1. The hyperbolic arctangent is defined up to ±∞, therefore Fahlman approximates it by using a value of −17.0 below −0.9999999 and +17.0 above +0.9999999. This error signal for output units behaves linearly for small differences between the actual output and the desired output, but for larger differences the error signal grows faster than linearly. This enables output units with larger output errors to change their weights faster to reduce the output error. Using the hyperbolic arctangent also avoids the problem of saturated neurons, because the derivation of a unit's transfer function is not needed to calculate the error signal.

After the training, the quality of a CNN is determined by a set of test vectors. The elements in this test set have the same structure as the training elements (but with the absence of the target output), but they are not included in the training set. The ability of a CNN to calculate the correct output from the presented input is called generalization. This means that the CNN is able to interpret in a reasonable way unknown data, which can be heavily deviant to the training data (e.g. data with noise).

In iterative inversion methods, the time consuming calculations of forward modelling have to be done for each inversion. In contrast, the numerically intensive calculations of CNNs appear only in the design phase (fixing the network arcitecture and preparing the training set and test set) and in the learning phase. After the learning stage, CNNs work at a very high processing speed due to the simple calculations required to determine the network output. While the training of the CNNs is being done by the ALPHA cluster at the University of Vienna, the application of the CNNs takes place on PC hardware.

4. CONCEPT

One difficulty in developing a detection system for 2D structures arises because of the flight path along the observed profile segment. It cannot be assumed that the sensor is maintained at a constant height h_0 across the entire profile segment. Because of the strong dependance of the signal's amplitude on the sensor height, sizeable changes in the measured values are created by variations in height. These alterations may hide or distort the anomalies caused by 2D structures. To prepare the CNN for these distortions, several flight paths over each single strucure should be presented to the network. Because of the time-consuming forward modelling of 2D strucures, it would be inefficient to do so.

Another difficulty is the large number of input parameters received by the CNN. As profile segments of length 1000 m have 101 measuring points (10 m point separation) and 8 values per measuring point, there are 808 resulting input values. Such a large number of inputs automatically increases the number of weights in the CNN and thereby slows down the training.

Based on many years of experience in the interpretation of AEM data, we used the following approach: 3 CNNs were first trained to calculate the parameters of 3 different homogeneous half-spaces (HHS). The first was based on the voltage ratios U_S/U_P for the frequencies 434 Hz and 3212 Hz; the second by frequencies 3212 Hz and 7002 Hz; and the third by frequencies 7002 Hz and 34133 Hz. The effects of the sensor height variations could be reduced by considering the resistivities of these HHS instead of the measured ratios U_S/U_P. The differing half-space resistivities caused by changing sensor heights did not pose much concern. It was shown that the CNN was able to produce a correct classification despite these differing resistivities. It was adequate, therefore, to train the CNN with a few constant sensor heights in order to detect 2D structures.

Another positive consequence of replacing the measured ratios U_S/U_P with the resistivities of the 3 homogeneous half-spaces was the reduction of the number of input parameters from 8 to 3 per measuring point. The number of input parameters was reduced from 808 to 183 by increasing the measuring point intervals with greater distance from the centre of the profile segment (a permissible stategy as the strongest variations in the measured values occurred at the center of the anomaly).

Based on previous considerations, the following method was applied (see Figure 14.4):

1. Division of the entire measured profile into segments of length 1 km (10 m intervals)
2. Calculation of the three different homogeneous half-spaces (CNN 1 to CNN 3) for the selected points of the profile segment
3. Using the specific resistivities of these three homogeneous half-spaces, the fourth CNN performs the detection of the 2D structures (The network output varies continuously between 0=not found and 1=found)
4. The final step was to calculate a binary output (structure found or not found) from the CNN output. With a successfully trained network, it was possible to control the detection reliability by using a threshold.

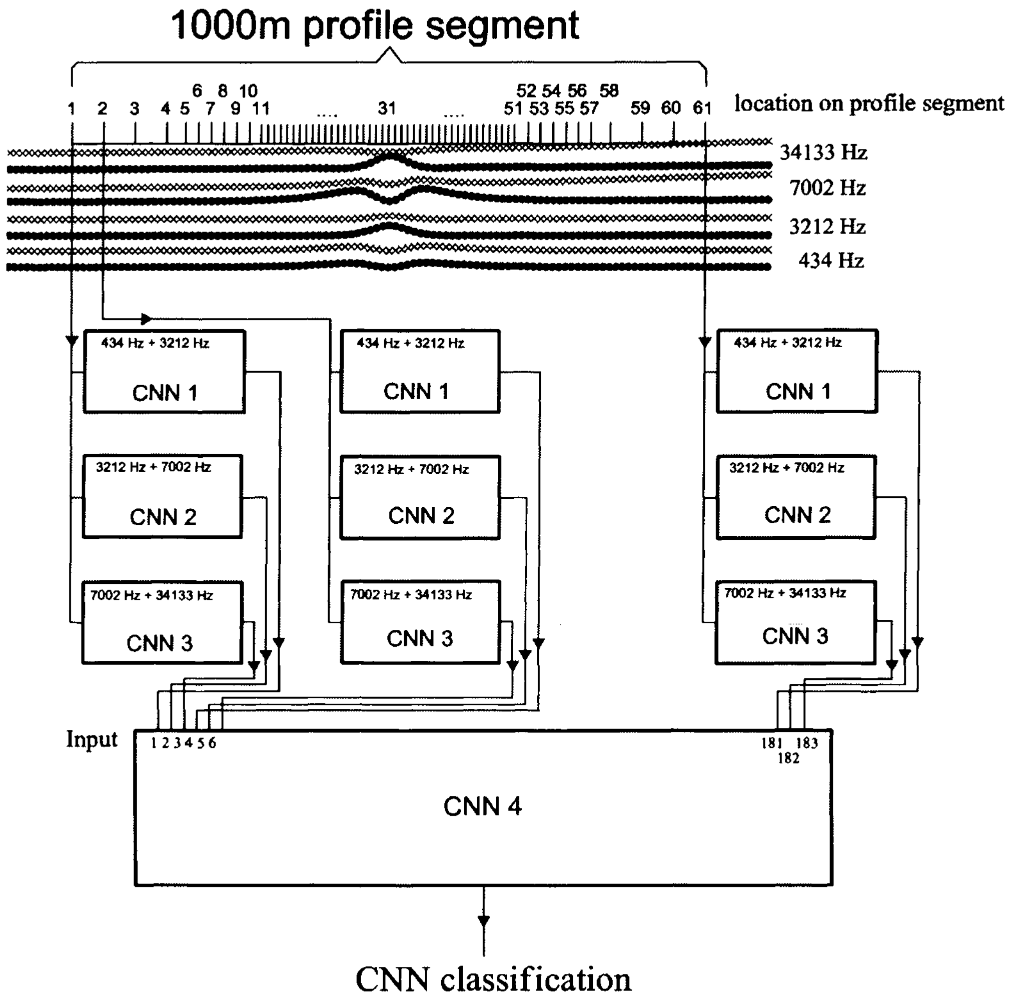

1000m profile segment
1 2 3 4 5 6 7 8 9 10 11 .... 31 .... 51 52 53 54 55 56 57 58 59 60 61
location on profile segment
34133 Hz
7002 Hz
3212 Hz
434 Hz
434 Hz + 3212 Hz
CNN 1
3212 Hz + 7002 Hz
CNN 2
7002 Hz + 34133 Hz
CNN 3
Input
1 2 3 4 5 6
181 182 183
CNN 4
CNN classification

Figure 14.4. Schematic presentation of the concept for the detection of 2D structures.
• ... Inphase x ... Outphase

5. CNNS TO CALCULATE HOMOGENEOUS HALF-SPACES

Three CNNs were trained to determine the parameters of homogeneous half-spaces corresponding to the complex voltage ratios U_S/U_P of two frequencies (frequency a and b) each (Table 14.3). The trainings vectors for these CNNs all have the same form,

$$(\underbrace{a_{IN}, a_{OUT}, b_{IN}, b_{OUT}}_{\text{INPUT}} \; / \; \underbrace{h_0, \rho_1}_{\text{TARGET OUTPUT}}),$$

where h_0 and ρ_1 are the parameters of the homogeneous half-spaces. For testing reasons only the INPUT values are used and the actual output is compared with the TARGET OUTPUT.

Table.14.3
Frequencies used to calculate corresponding homogeneous half-spaces

	Frequency a	Frequency b
CNN 1	434 Hz $\Rightarrow f_1$	3212 Hz $\Rightarrow f_2$
CNN 2	3212 Hz $\Rightarrow f_2$	7002 Hz $\Rightarrow f_3$
CNN 3	7002 Hz $\Rightarrow f_3$	34133 Hz $\Rightarrow f_4$

A critical point in training a CNN is in the selection of the training set. By using too many training samples or by using wrong samples, successful training can be delayed, or might even be impossible. We calculated the theoretical complex voltage ratio U_S/U_P (see Section 2) for a class of homogeneous half-spaces. In this training set we checked if similar input patterns represent half-space models with heavily varying model parameters (cluster analysis). Such training samples were summarized in one cluster and replaced by the training vector nearest to the cluster centroid. The training of the CNNs to determine the parameters of HHS (CNN 1 to 3) was done with the number of training samples given in Table 14.4.

Table 14.4
Range of variation for the parameters of HHS to calculate the training set and number of training samples used for the training of each CNN (see Table 14.3).

	Specific resistivity	Sensor height	Number of training samples
CNN 1	1 to 1000 Ωm	30 to 150 m	223
CNN 2	1 to 6000 Ωm	30 to 150 m	224
CNN 3	1 to 6000 Ωm	30 to 150 m	316

For these CNNs, the same network architecture was used (Figure14.5)

- One input layer with 4 input neurons (plus one bias neuron)
- One hidden layer with two different types of neurons (Table.14.6). From each type of transfer function there is the same number of neurons.
- One output layer with 2 output neurons. In this layer, cubic_sigmoid_11 transfer functions were used (Section 3).
- Each neuron is connected to each other neuron in following layers.

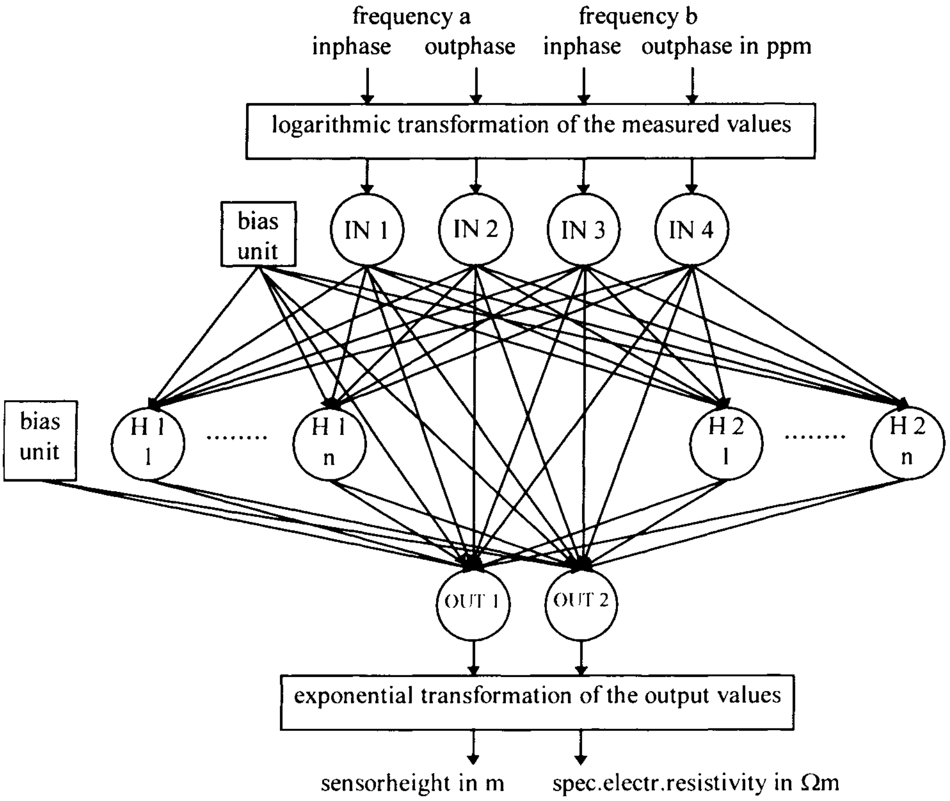

frequency a
frequency b
inphase
outphase
inphase
outphase in ppm
logarithmic transformation of the measured values
bias unit
IN 1
IN 2
IN 3
IN 4
bias unit
H1 1
....
H1 n
H2 1
....
H2 n
OUT 1
OUT 2
exponential transformation of the output values
sensorheight in m
spec.electr.resistivity in Ωm

Figure 14.5. Architecture of the CNN's to calculate the parameters of homogeneous half-spaces. IN ... input neuron; H1, H2 ... hidden neuron with transfer function of type 1 resp. 2; (see Table 14.6); OUT ... output neuron.

After successfully training, the various networks were tested with a large number of synthetic homogeneous half-space models (see Table14.5).

Table 14.5
Range of variation for the parameters of homogeneous half-spaces to calculate the test set, and number of test samples used to test each CNN (see Table 14.3).

	Specific resistivity	Sensor height	Number of test samples
CNN 1	1 to 1000 Ωm	29 to 151 m	32612
CNN 2	1 to 6000 Ωm	30,5 to 150,5 m	35746
CNN 3	1 to 6000 Ωm	29 to 151 m	22816

In these tests, the absolute values of the differences between the CNN results (calculated h_{0_CNN} and $\rho_{1\ CNN}$) and the parameters h_0 und ρ_1 of the models in the test sets were calculated. Based upon these tests, one CNN per frequency pair (see Table14.3) was chosen. In making these decisions, it was important that the CNNs show good performance at a sensor height of about 50 m (the most common sensor height). The number and the types of neurons, used in the hidden layer of the neural networks CNN 1 to CNN 3, are summarized in Table 14.6.

Table 14.6
Number and types (transfer function) of the neurons in the hidden layer of the neural networks CNN 1, 2 and 3 (because of transfer functions see Section 3 of this chapter).

	CNN 1 (f_1, f_2)	CNN 2 (f_2, f_3)	CNN (f_3, f_4)
Number of neurons in the hidden layer	15+15	14+14	9+9
Types of transfer functions in the hidden layer (see Fig.14.5)	elliot_sigmoid_11 + hyperbolic_ tangent_11	elliot_sigmoid_11 + hypsigmoid_11	elliot_sigmoid_11 + hypsigmoid_11

6. CNN FOR DETECTING 2D STRUCTURES

6.1. Training and Test vectors

To train or test the CNN to detect 2D structures, a large number of synthetic AEM profiles have to be calculated (see Table14.7). The model categories 1, 2, 3, 6, 7 and 8 have been calculated with the aid of the theory discussed in Section 2 (forward modeling for 1 dimensional models). The calculations for the model categories 4 and 5 have taken place with the program EMIGMA (see Section 2 of this chapter).

Table 14.7
AEM profiles with length 1000 m. In the model categories 4 and 5 the 2D structure is in the center of the profile segment (see Figure14.6).

Model category	Number	Remarks
1. Homogeneous half-space (HHS)	27	ρ_1=1-10000 Ωm ; h_0 = 40,55,70 m
2. HHS with ±1ppm error	27	
3. Homogeneous 2 layer half-space models (2LHS)	144	ρ_1=1- 000 Ωm ; h_0 = 40,55,70 m ρ_2=1- 000 Ωm ; h_1 =1-31,6 m (see Fig.14.1)
4. Vertical contact	168	ρ_1=1-10000 Ωm ; h_0 = 40,55,70 m ρ_2=1-10000 Ωm (see Fig.14.6)
5. Dike	192	h_0 = 40,55,70 m ; d = 2, 5, 10, 20 m t = 2, 5, 10, 20 m ; ρ_1 = 1000 Ωm ; σ·t-product = 10, 30, 100, 300S (see Fig.14.6)
6. 2 layer random models	200	see Section 6.3
7. Symmetrical profiles of 3 layer random models	100	see Section 6.3
8. 3 layer random models	100	see Section 6.3
Total	958	

In addition, profiles with a length of 2000 m were calculated for the 192 dike structures (dike is in the center of the profile, see Table 14.7). To each of these AEM anomalies, an error term of ±1 ppm and ±2 ppm was added. Therefore, there are a total of 576 profiles with a length of 2000 m.

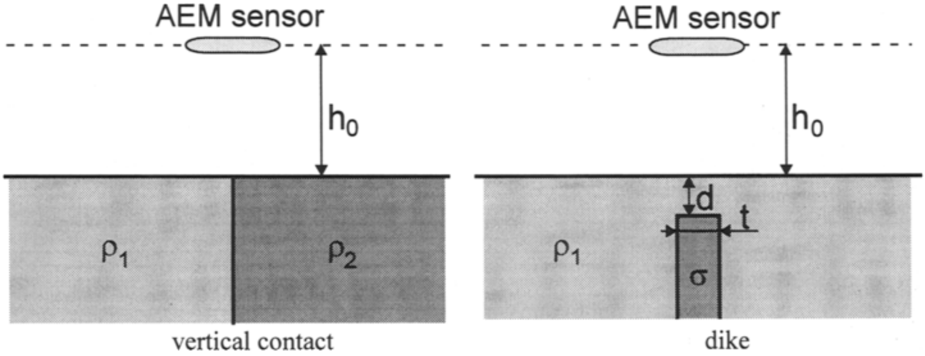


Figure 14.6. Schematic picture of the calculated 2D structures (see Table14.8).

6.2. Calculation of the error term (±1ppm, ±2ppm)

To calculate the synthetic error for a profile, an initial random error between –1 ppm and +1 ppm (or –2 ppm and +2 ppm) was calculated for the first profile position. For the error values of the next position, only the changes to the current error value (±0.2 ppm) were determined randomly. If an error value would exceed the error range when the error change was applied, the change is added with reversed sign. In this way an error term is obtained which a simple low pass filter cannot suppress.

6.3. Calculation of the random models (model categories 6-8)

The first step in calculating the random models was to determine the random parameters of 2 layer half-space models (or 3 layer half-space models). At first the parameters of an initial layered half-space model were calculated randomly. To determine the following model parameters on the profile, only the changes in the parameters were calculated randomly. In the case of the symmetric profiles of random models (see model category 7), the model parameters were only determined as far as the centre of the profile and were then mirrored around the center. Using these model parameters, the corresponding complex voltage ratios U_S/U_P for layered half-space models were calculated for each point of the profiles.

6.4. Training

For the CNN to detect the 2D structures, the following network architecture was chosen:

- One input layer with 183 input neurons (plus a bias-neuron)
- Two hidden layers with neurons using hypsigmoid_11 transfer functions (see Section 3).
- One output layer with a neuron using a cubic_sigmoid_01 transfer function (see Section 3).
- Each neuron is connected to each neuron in all following layers.

During the training of the CNN, the number of neurons in the hidden layers was varied from 1 to 4.

The training vectors for this CNN all have the same form

$$(\underbrace{\rho_{1\_CNN1}, \dots, \rho_{61\_CNN1}, \rho_{1\_CNN2}, \dots, \rho_{61\_CNN2}, \rho_{1\_CNN3}, \dots, \rho_{61\_CNN3}}_{\text{INPUT}} \,/\, \underbrace{\text{classification}}_{\substack{\text{TARGET}\\ \text{OUTPUT}}}),$$

where ρ_{m_CNNn} is the resistivity of the n-th homogeneous half-space on the m-th selected point of the profile segment (see Section 5). In case of a dike structure, the output was 0.9 otherwise 0.1.

From the 958 available models (Table 14.7) a number of vectors were selected for training (Table 14.8).

Table 14.8
Number of models used for training

Model category	1	2	3	4	5	6	7	8	Total
Number	6	11	21	66	64 (⇒* 2 x 64)	113	22	18	385

* The selected vectors of dike models were used twice to balance the training set.

The selection of the trainings vectors was conducted by considering a cluster analysis. As one can see in the selection (Table 14.8), only profiles with the dike structures in the center of the profile segment were used to train the CNN. To prevent the CNN from detecting only the symmetry of the signal, 22 symmetrical profiles of 3 layer random models were also used for training (model category 7).

Based on extensive tests, a CNN with 3 neurons in the 1st hidden layer and 1 neuron in the 2nd hidden layer (plus one bias neuron in each hidden layer) was finally selected to perform the detection of 2D structures.

7. TESTING

At first, the 637 profiles with length 1000 m (Table 14.7), which are not contained in the training set, were interpreted by the CNNs 1 to 4 (Figure 14.4). This means that three different homogeneous half-spaces were calculated using the networks CNN 1 to CNN 3. Based on the resistivities of these three homogeneous half-spaces, the detection is performed by CNN 4.

As one can see in Figure 14.7, every profile that shows no dike structure produced a network output of less than 0.2. In contrast each distinct dike anomaly caused a network output greater than 0.8. The profiles named as 'INDISTINCT DIKE' represent models of dike structures producing very small anomalies (the mean value of the anomalies of all 8 measured components are less then 3 ppm). For these dike models a correct classification cannot be expected. For this reason these dikes were not used for the training. None of the symmetrical profiles of 3 layer random models (model category 7 - Table 14.7) were accidentally classified as a dike.

The AEM responses of all 192 dike models were calculated for profile segments of length 2 km (10 m intervals). For every profile the dike was located at the center of the profile segment. These profiles were used to examine the answers produced by the CNN when the dike anomaly is not located at the center of the profile segment considered by the CNN. To each of these AEM anomalies, an error term of ±1 ppm and ±2 ppm was added. The procedure by which the error was determined is explained in Section 6. In this way 576 test profiles of length 2000 m were produced.

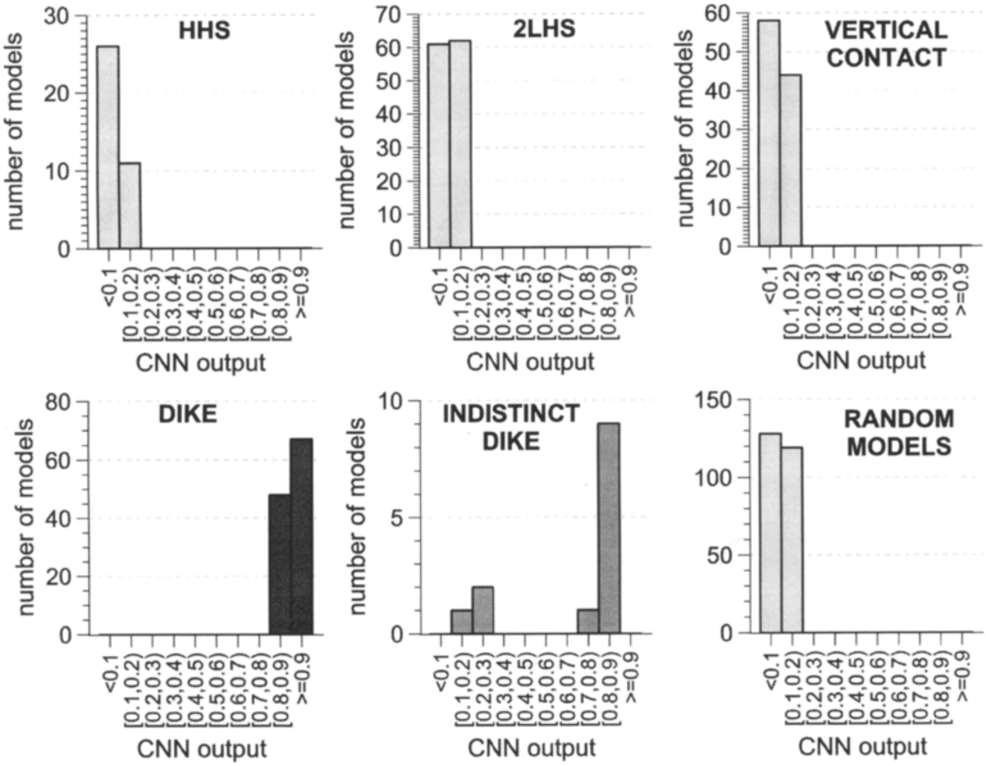


Figure 14.7. Results of the CNNs on 637 profiles with length 1000 m (Table 14.7). INDISTINCT DIKE is a dike structure with very weak anomalies.

Based on the complex voltage ratios U_S/U_P, three different HHS were calculated by the networks CNN 1 to CNN 3. The 2km long profiles, with the three resistivities, were divided into 101 profile segments each of length 1km. This means that the dike is always located at a different position within these 1000 m profile segments. Each of these profile segments was then classified by CNN 4. The classification result for the profile segment is assigned to the coordinates of the segment center. In doing so, the result is to obtain a profile of length 1000 m with the classifications given by the CNN.

To detect the start, center and end of the area classified as a dike, the changes of the network output are monitored as a function of the position of the profile (horizontal gradient). The start and end positions of the area classified as a dike are recorded when the absolute value of the gradient exceeds a specified threshold value. The actual position of the dike is then assumed to be located midway between the two marks (Figures 14.8 to 14.10). A summary of the results is given in Table 14.9.

Table 14.9
Results of the tests for detecting dikes from 576 profiles with 2000m lengths.

	No noise	±1ppm noise	±2ppm noise
Number of dikes successfully detected	188 (97.9%)	170 (88.5%)	122 (63.5%)
Number of dikes undetected	4	22	70
Mean shift between calculated position and true postion of detected dikes	-6.9±4.5 m	-9.3±22.4 m	-10.5±24.6 m
Maximum shift between calculated position and true postion of detected dikes	15 m	65 m	130 m
Number of dikes detected where none exist	0	0	0

The figures 14.8 and 14.9 are examples of the network outputs and the determined location of the dikes for dike anomalies without noise (Figure 14.8) and with ±2 ppm noise (Figure 14.9).

In addition, the complex voltage ratios U_S/U_P for the same dike structure were calculated for a descent of the sensor height from h_0=80 m to h_0=30 m. Noise of ±2 ppm was added to this profile. Figure 14.10 shows the network output and the estimated location of the dike for this profile.

8. CONCLUSION

The aim of our work was to develop an automatic detection of geophysical 2D structures with the aid of airborne electromagnetic (AEM) data along a profile segment. The 2D structure was chosen to be a dike because of its relevance in mineral exploration.

It is the authors' opinion that iterative methods are not suitable because of the time-consuming nature of the forward modeling of 2D structures. As an appropriate alternative, we decided to use CNNs to develop the automatic detection. This technique has the advantage that trained CNN can perform computations at great speed on normal PC hardware. The time consuming training of the CNN, which must be done only once for a specific sensor, can be done efficiently on high performance computers.

The chosen concept (see Section 4) provides the training for 4 different CNNs. Three of these CNNs are used to calculate the parameters of three different HHS with two of the four frequencies selected consecutively. Based on the resistivities of these HHS, the classification of the profile data is determined (see Section 6).

Tests on these trained CNNs were performed with synthetic data (see Section 7). As the new sensor the classification is planned for was not yet in use at the time of this work, we were not able to test the network with real data. On the basis of our experience, the CNN technique is an efficient tool for interpreting AEM data. In particular, the automatic interpretation of the AEM data in the context of 2 or 3 dimensional subsurface models appears to be a very practical application of CNNs.

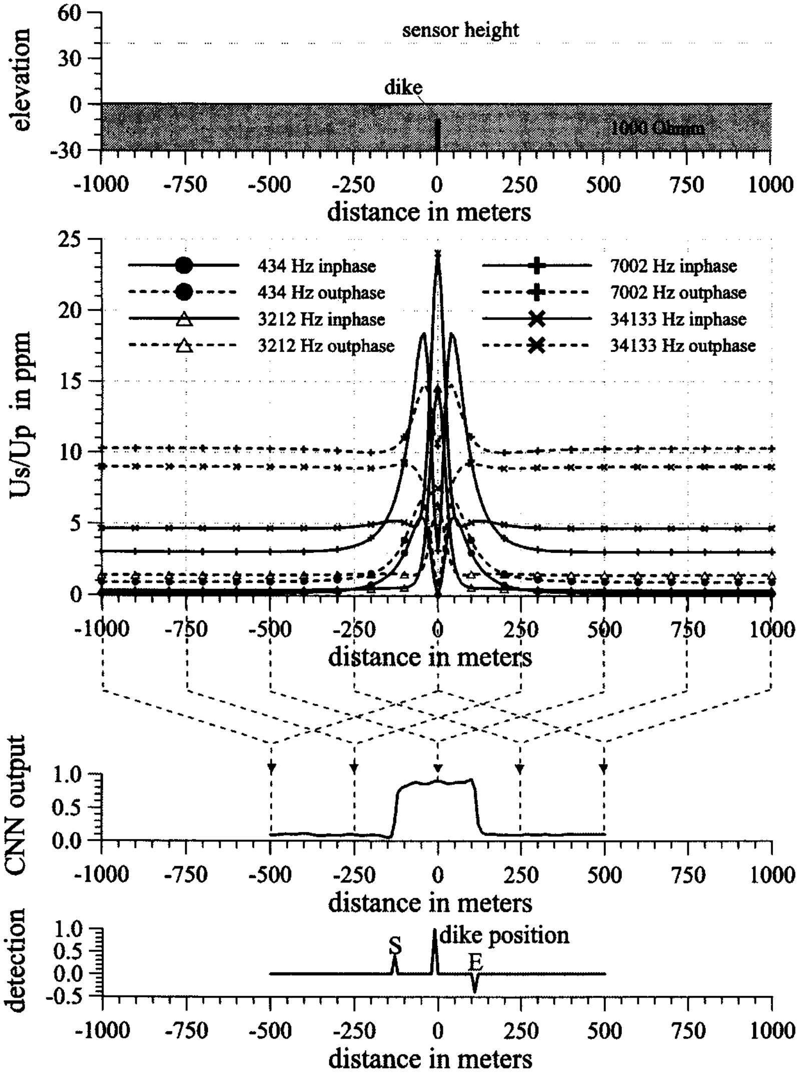


Figure 14.8. Network output and detected location of a dike structure (h_0=40 m, d=10 m, t=10 m, σ=3 S/m). Detection based on synthetic anomalies without noise (mean value of the anomalies of all 8 components is 9 ppm). S defines the start and E the end of the area classified as a dike.

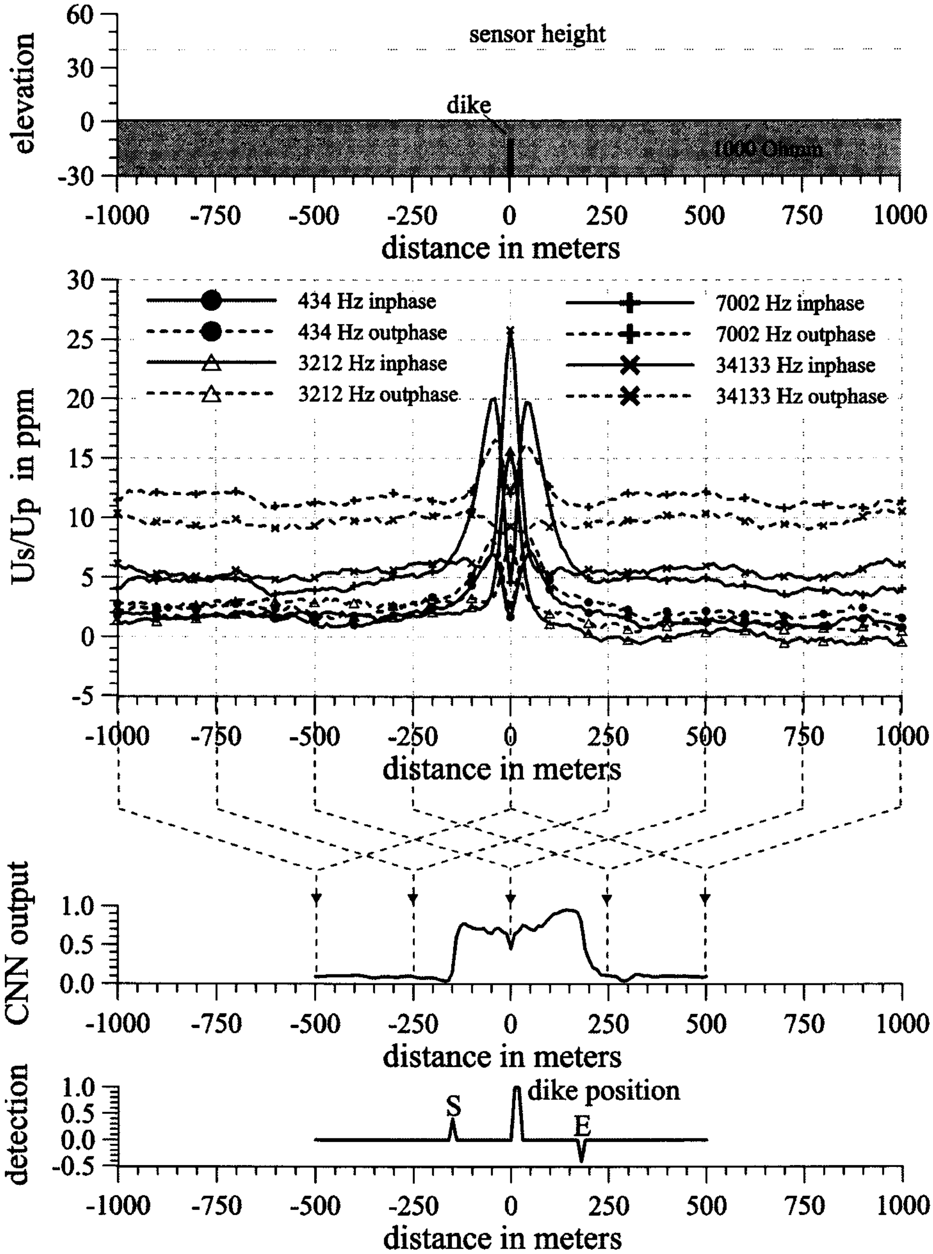


Figure 14.9. Network output and detected location of a dike structure (h_0=40 m, d=10 m, t=10 m, σ=3 S/m). Detection based on synthetic anomalies with ±2 ppm noise (mean value of the anomalies of all 8 components is 9 ppm). S defines the start and E the end of the area classified as a dike.

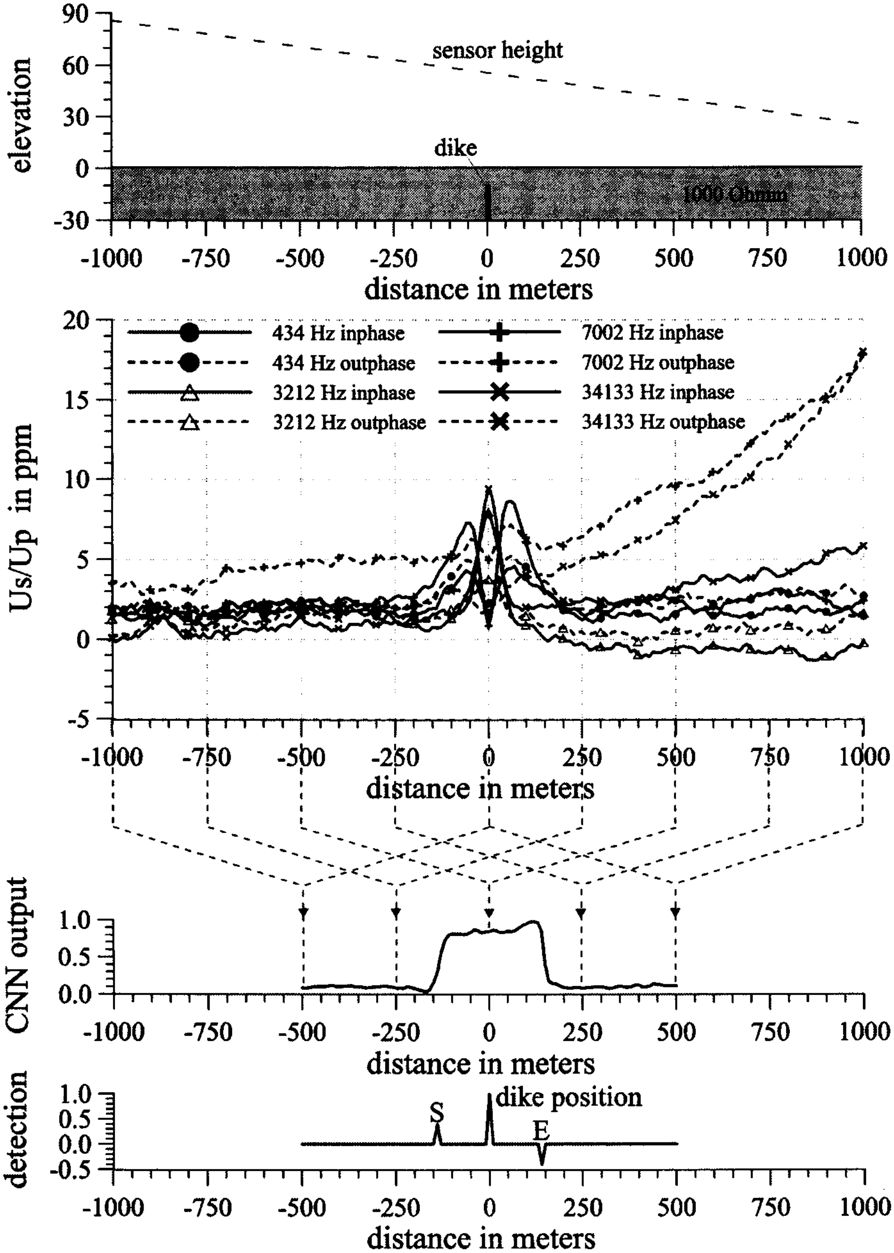


Figure 14.10. Network output and detected location of a dike structure (h_0=80→30m, d=10 m, t=10 m, σ=3 S/m). Detection based on synthetic anomalies with ±2 ppm noise. S defines the start and E the end of the area classified as a dike.

REFERENCES

Anguita, D., Parodi, G., and Zunino, R., 1993, Speed improvement of the back-propagation on current generation workstations, *in* Proceedings of the World Congress on Neural Networking, Portland, Oregon, Lawrence Erlbaum/INNS Press, **1,** 165-168.

Elliott, D., 1993, A better activation function for artificial neural networks: Technical Report TR 93-8, Institute for Systems Research, University of Maryland.

Fahlman, S., 1988, An empirical study of learning speed in back-propagation networks: report CMU-CS-88-162, September 1988.

Fraser, D., 1978, Resistivity mapping with an airborne multicoil electromagnetic system: Geophysics, **43**, 144 - 172.

Harrington, R., 1968, Field Computation by Moment Methods: Macmillan.

Hohmann, G., 1975, Three-dimensional induced polarization and electromagnetic modelling: Geophysics, **40**, 309-324.

Petros Eikon, Inc., 1997, Forward 3-D electromagnetic simulation platform for comprehensive geophysical modeling, manual for software release version 5.15, EMIGMA/V5.15, September 1997

Sengpiel, K., 1988, Approximate inversion of airborne EM data: Geophysical Prospecting, **36**, 446 - 459.

Sontag, E., 1990, On the recognition capabilities of feedforward nets: Department of Mathematics, Rutgers University, New Brunswick, NJ.

Wait, J., 1982, Geo-Electromagnetism: Academic Press.

Chapter 15
Locating Layer Boundaries with Unfocused Resistivity Tools

Lin Zhang

1. INTRODUCTION

Resistivity tools are used for determining lithology, locating layer boundaries, and estimating invasion and formation resistivities. Resistivity tools may be focused or unfocused. The unfocused tool data are difficult to extract layer boundaries from so I have explored the applicability of using neural networks for this problem. I extract layer boundaries using data from three unfocused tools with different depth of investigations and compare several different network learning algorithms to see which produces the most accurate layer boundaries.

The earliest unfocused resistivity tool is called a normal tool. The tool has a current electrode *A* (transmitter) and a measurement electrode *M* (receiver). The distance between the electrodes *A* and *M* is typically: 20.3 cm (8 in.), 25.4 cm (10 in.), 40.6 cm (16 in.), 45.7 cm (18 in.), 81.3 cm (32 in.), 96.5 cm (38 in.), 99.1 cm (39 in.), 160.0 cm (63 in.), 162.6 cm (64 in.), and 213.4 cm (84 in.). The AM spacing of the "short normal" curve historically has been 16 in. and the "long normal" electrode configuration usually has the AM spacing 162.6 cm (64 in.).

There are two problems related to the unfocused resistivity tools. In a borehole with very conductive mud, the current tends to flow in the mud instead of the formation. Therefore, the apparent resistivity must be deduced from the injected current, and the resultant voltage will not accurately reflect the formation resistivity (Ellis, 1987). The second problem is that the conductive mud provides an easy path into adjacent shoulder beds of much lower resistivity. Therefore, the apparent resistivity will not represent the resistive bed, but more likely, the less resistive shoulder bed.

The main difference between a focused measurement and an unfocused measurement is the distribution of the current emitted from electrode *A*. Figure 15.1 shows a schematic drawing of focused and unfocused electrical current distribution about a logging tool. R_t is the formation resistivity in a resistive bed and R_S is the formation resistivity in the adjacent layer, which is very conductive. The left-hand side of Figure15.1 is for the unfocused system. The current lines diverge from A in all directions and are attracted upward and downward by the adjacent formations, which are more conductive than the layer, so that the resistance offered by the layer to the current is, to a great extent, reduced. The apparent resistivity read opposite the layer is, therefore, much lower than the 'true resistivity' of the layer. The right hand side of the Figure 15.1 is for a focused system; contiguous guard currents focus all the current lines. Focused currents are less prone to borehole effects, especially the conductive mud effects, and can be directed at required areas of the formation. The apparent resistivity then is very close to the true resistivity of the layers.

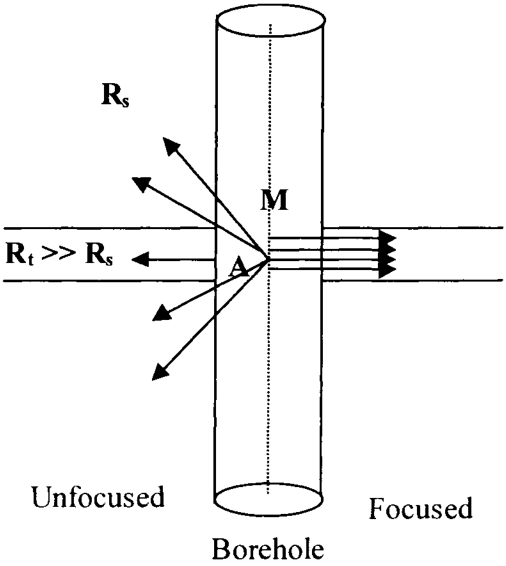


Figure 15.1. Schematic drawing of focused and unfocused electrical current distribution around the logging tool.

The unfocused resistivity tools have been widely used in many areas of the world. For sedimentary deposits, such as coal, the unfocused resistivity tools have an important role in qualitative and quantitative interpretation. With computerized interpretation, Whitman (1995) found that the unfocused resistivity tools have much better vertical resolution and generally higher quality information on formation resistivities than previously believed.

There are some important characteristics of the unfocused measurement:

1. The shallow unfocused device (short normal) is greatly affected by invasion; thus it cannot, in general, show the true resistivities. The fact that it closely reflects the resistivity of the invaded zone makes it a useful tool to estimate the effect of invasion.
2. The deep unfocused measurement (long normal) is not well adapted to the definition of thin layer boundaries but is sufficient for finding R_t in thick layers.
3. The unfocused measurement tends to show resistive beds as thinner than they actually are by an amount equal to the spacing, and they tend to show conductive layers thicker than they actually are by an amount equal to the spacing. (See Figure 15.2.)
4. For thin, resistive layers, the surrounding formations appear on the logs as being conductive. The more resistive they are, the more conductive they appear.

The tools I used in this study, L045, L105 and L225 are unfocused resistivity tools developed in Russia. The characteristics of these tools are listed in Table 15.1.

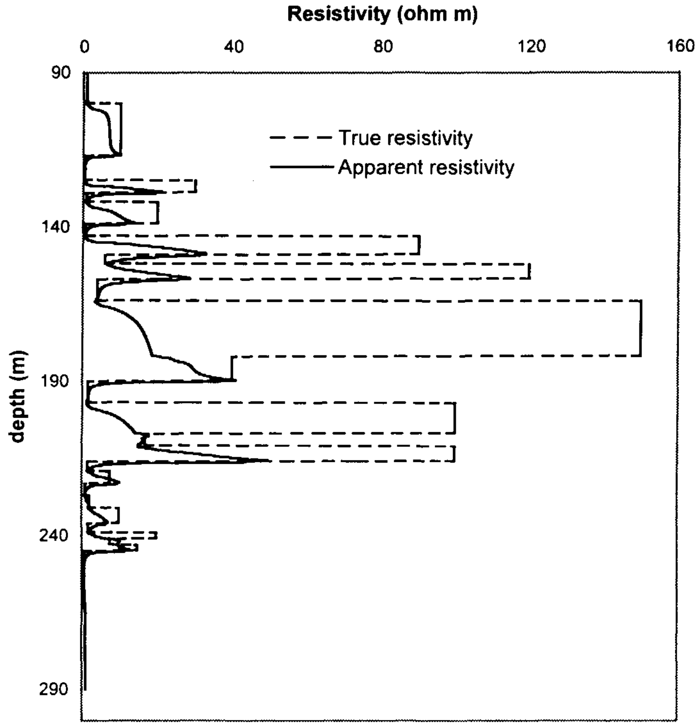


Figure 15.2. Apparent resistivity measured by a shallow unfocused tool. The conductive layers are shown thicker than they actually are and the resistive layers are shown thinner than they actually are.

Table 15.1
Characteristics of the Russian unfocused tools

Log name	AM spacing (m)	Depth of investigation	Minimum bed resolution (m)
L045	0.45	Shallow	0.5
L105	1.05	Deep	1.0
L225	2.25	Deep	2.0

2. LAYER BOUNDARY PICKING

Layer boundaries in geology are generally defined as a distinctive, planar unit limited by the significant differences in lithology, composition, or facies, etc. (Rider, 1996). The layer boundaries can provide important information for well logging interpretation.

The goal of log interpretation is to determine the physical boundaries in the subsurface based on the changes in rock or fluid properties. The best geophysical logs for determinating the boundaries are those with a moderate depth of investigation, SFL (spherically focused logs) and density logs (Rider, 1996), but those tools do not run in every well or every section of a well. The conventional rule to pick the layer boundaries is based on the mid-point of the tangent to a shoulder. This is an identifiable method and can be applied consistently under isotropic conditions. Under anisotropy conditions, however, the method cannot provide an accurate position of the layer boundaries. Thus, the experienced log analyst must use the changes from several log properties to indicate the boundaries. However, there are some shortcomings: 1) personal judgment used to pick boundaries from well logs may not provide reliable results; 2) two log analysts may have different criteria for choosing the boundaries, hence, there might be different results for the same group of log data; 3) picking boundaries in a large data set can be very time-consuming and tedious.

For a focused resistivity tool, the layer boundaries are chosen based on the inflection points, maximum change in slope, etc. For an unfocused logging tool, the unfocused effects can shift the log response and the layer boundaries may not coincide with inflection points.

The layer boundary and resistivity from an unfocused resistivity tool can be estimated from inversion (Yang and Ward, 1984). The authors reported on an investigation of the inversion of borehole normal (unfocused) resistivity data. Interpretation included individual thin beds and complicated layered structure using the ridge regression method. A ridge regression estimator has both the gradient method, which is slow but stable to converge, and the Newton-Raphson technique, which is fast but may be divergent. The forward model contained an arbitrary number of layers. Two forward model results for resistive and conductive thin beds indicated the difference between the true resistivity and apparent resistivity are affected by the distance between source A and electrode M. In other words, the smaller the distance between transmitter and receiver, the better the resolution of the thin bed. The synthetic model results and the field examples indicated that the inverse method could be used to estimate layer thickness and resistivity.

Whitman et al. (1989) investigated a 1D and 2D inversion of unfocused and focused log responses for both synthetic logs and field data. The ridge regression procedure (Marquardt's inversion) is applied to solve the inverse problems to determine the earth parameters, such as layer boundaries, invasion zone, mud resistivity, and vertical and horizontal resistivity distribution from unfocused and focused resistivity logs. The method was tested on synthetic and field data for the 40.6 cm (16 in.) and 162.6 cm (64 in.) unfocused resistivity log, as well as for the 5.5 m (18 ft.) and 20.3 cm (8 in.) focused resistivity log. The results indicated that the initial guess model determined the quality of the final model.

An automatic inversion was developed by Whitman et al. (1990) to invert the responses of the unfocused resistivity tools and to interpret data from these logging tools for bed boundaries and formation resistivity. From the previous research (Whitman et al., 1989), inversion is largely dependent on the initial model. Thus, the choice of the initial model is very important, but usually done by hand. The authors show how to automatically choose the initial model parameters (thickness and resistivity) through the construction of an approximate focused log from the original normal log.

To pick the layer boundaries and resistivities for the initial model, an approximate focused log (Laterolog 7) was generated from the measured unfocused log. Rationale for the approach is that the focused log has a better definition of the layer boundary and true bed resistivity. The layer boundaries are chosen by the relatively abrupt changes in the ratio of the two focusing currents. The corresponding bed resistivities are then picked directly from the synthetic focused log. The basic theory for the Laterolog 7 is given by Roy (1981), showing that the response of a focused resistivity log could be simulated by unfocussed resistivity logs having different spacing. Based on this principle, the focused logs could be calculated from the unfocused resistivity logs.

Once the initial model has been chosen, finite difference estimation to the potential equation is used in the forward modeling. The inversion procedure follows the ridge regression procedure (Marquardt's inversion) and the ill-conditioned matrices are avoided by the stabilizing parameters.

The inversion results from two unfocused resistivity logs are compared between the automatic initial model and hand picked initial model. The results show that the automatic initial model performs quite well, indicating the automatic procedure performed at least as well as that using hand picks for the initial guess model.

Whitman (1995) pointed out that interpretation of unfocused resistivity logs is relatively easy when the bed thickness is at least 1.5 times the tool spacing. When the bed thickness is less than this, determination of the correct Rt for these beds will be difficult because nearby beds can substantially affect the apparent resistivity measured by the log.

To solve this problem, inversion software was developed with a built-in initial guess function that makes an automatic initial guess of bed boundaries and true formation resistivity (Whitman, et al., 1989). The inversion follows the Levenberg-Marquardt procedures to minimize the root-mean-square (RMS) between the field log and the simulated log. After inversion, the overlay of the associated earth models can be used to indicate the invasion zone, impermeable zones, gas/water and oil/water contacts, and layer boundaries with a resolution of 0.61 m (2 ft.) to 0.91 m (3 ft.). The Oklahoma Benchmark earth model was used to test this inversion program. The results are consistent and reliable. However, the author indicated that inversion of a 500 ft unfocused log on an IBM RS6000 model 550 requires at least eight hours of CPU time.

In recent years, neural networks have been applied to solve various geophysical problems. The traditional layer picking method based on the maximum change in slope was difficult in the presence of noise and in thin-bed regions so Chakravarthy et al. (1999) applied neural networks to the detection of layer boundaries from the High Definition Induction Log (HDIL). A radial basis function network (RBFN) was implemented. The HDIL is a multi-receiver, multi-frequency induction device that measures formation resistivities at multiple depths of investigation (Beard, et al., 1996). Synthetic responses for seven subarrays, which have a large range of spacing from 15.2 cm (6 in.) to 2.4 m (94 in.) and eight frequencies, which range from 10 kHz to 150 kHz, are generated for varying ranges of thickness, invasion length, formation resistivity, and invasion zone resistivity. The synthetic data along with the true bed boundary locations are used to train the neural network for picking layer boundaries. The logarithmic derivative of the log data was computed first; secondly, the transformed logs were broken into overlapping sections of fixed length. Data in each section or window were normalized to a unit norm. Thirdly, the normalized sections were presented to the neural network as training patterns. If the center of the training pattern corresponded to the boundary, then output 1, otherwise, output 0. The RBFN was successfully applied to the Oklahoma Benchmark model and Gulf of Mexico HDIL data to delineate layer boundaries. It demonstrated that the neural networks have the ability to detect layer boundaries.

Little work has been done on the interpretation of unfocused resistivity responses using neural networks. Thus, a neural network based method for picking layer boundaries from the unfocused resistivity logs has been developed and is described next.

3. MODULAR NEURAL NETWORK

The modular neural network (MNN) consists of a group of modules (local experts) and a gating network. The network combines supervised and unsupervised learning. The gating network learns to break a task into several parts, which is unsupervised learning, and each module is assigned to learn one part of the task, which is supervised learning.

Figure 15.3 is the block diagram of a modular network (Haykin, 1994). Both the modules and the gating network are fully connected to the input layer. The number of output nodes in the gating network equals the number of modules. The output of each module is connected to the output layer. The output values of the gating network have been normalized to sum to 1.0 (equation 15.1). These normalized output values are used to weight the output vector from the corresponding modules so the output from the best module will be passed to the output layer with little change while the outputs from the other modules will be weighted by a number close to zero and will have little impact on the solution. The final output is the sum of the weighted output vectors (equation 15.2).

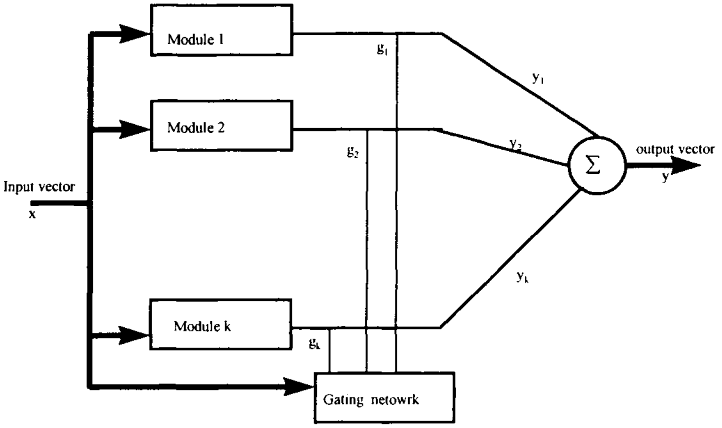


Figure 15.3. Block diagram of a modular network; the outputs of the modules are mediated by the gating network (Haykin, 1994).

The variables used in a MNN are defined as:

K: number of modules, also the number of output nodes in gating network
N: number of output nodes in MNN output layer and each module's output layer
M: number of input nodes
Q: number of hidden nodes in each module
P: number of hidden nodes in the gating network
$\vec{x}$ = $(x_1, x_2, \ldots\ldots, x_M)$
= Input training vector
$\vec{d}$ = $(d_1, d_2, \ldots\ldots, d_N)$
= Desired output vector
$\vec{u}$ = $(u_1, u_2, \ldots\ldots, u_K)$
= Output vector of the gating network before normalized to sum to 1
$\vec{g}$ = $(g_1, g_2, \ldots\ldots, g_K)$
= Output vector of the gating network after normalized to sum to1
$\vec{z}$ = $(z_1, z_2, \ldots\ldots, z_N)$
= Output vector of the whole network
$\vec{o}_k$ = $(o_1, o_2, \ldots\ldots, o_N)$
= Output vector of the k^{th} module
w^k_{qm} = Connection weight between hidden and output layer in k^{th} module
w^g_{pm} = Connection weight between hidden and input layer in the gating network
Sum^k_n =Weighted sum for PE n in module k

Each module or local expert and the gating network receive the same input pattern from the training set. The gating network and the modules are trained simultaneously. The gating network determines which local expert produced the most accurate response to the training pattern and the connection weights in that module are allowed to be updated to increase the probability that that module will respond best to similar input patterns.

The learning algorithm can be summarized as follows:

1. Initialization: Assign initial random values to the connection weights in the modules and the gating network.
2. Calculate the output for module k

$$o_n^k = f(Sum_n^k), \tag{15.1}$$

where $$Sum_n^k = \sum_{q=1}^{Q} (f(\sum_{m=1}^{M} x_m w_{qm}^k)) w_{nq}^k \tag{15.2}$$

3. Calculate the activation for the gating network

$$u_k = f(\sum_{k=1}^{K} (f(\sum_{p=1}^{P} x_m w_{pm}^g)) w_{kp}^g). \tag{15.3}$$

4. Calculate the softmax output for the gating network

$$g_i = \frac{\exp(u_k)}{\sum_{i=1}^{K} \exp(u_i)}. \tag{15.4}$$

5. Calculate the network output

$$z_n = \sum_{i=1}^{K} g_i o_i. \tag{15.5}$$

6. Calculate the associative Gaussian mixture model for each output PE

$$h_k = \frac{g_k \exp(\frac{-1}{2} \| \vec{d} - \vec{o}^k \|^2)}{\sum_{j=1}^{K} g_j \exp(\frac{-1}{2} \| \vec{d} - \vec{o}^j \|^2)}. \tag{15.6}$$

7. Calculate the errors between the desired output and the each module's output.

$$e_n^k = d_n^k - o_n^k. \tag{15.7}$$

8. Update weights for each module

Weights between the output and hidden layer:

$$w_{nq}^k(t+1) = w_{nq}^k(t) + \eta h_k \delta_{nq}^k act_q, \tag{15.8}$$

where $\delta^k_{nq} = e^k_n f'(Sum^k_n)$ and $act_q = f(\sum_{q=1}^{Q} x_m w^k_{qm})$. (15.9)

Weights between the input and the hidden layer:

$$w^k_{qm}(t+1) = w^k_{qm}(t) + \eta h_k \delta^k_{qm} x_m, \qquad (15.10)$$

where $\delta^k_{qm} = f'(Sum^k_q)\sum_{n=1}^{N} \delta^k_{nq} w$. (15.11)

8. Update weights for gating network

Weights between the output and hidden layer:

$$w^g_{kp}(t+1) = w^g_{kp}(t) + \eta \delta^g_{kp} act_p, \qquad (15.12)$$

where $\delta^g_{kp} = (h_k - g_k) f'(u_k)$ and $act_p = f(\sum x_m w^g_{pm})$. (15.13)

Weights between the input and the hidden layer:

$$w^g_{pm}(t+1) = w^g_{pm}(t) + \eta \delta^g_{pm} x_m,$$

where $\delta^g_{pm} = f'(Sum^g_p)\sum_{k=1}^{K} \delta^g_{kp} w^g_{kp}$.

4. TRAINING WITH MULTIPLE LOGGING TOOLS

The modular neural network has been trained and tested with data from multiple logging tools. Each tool requires forty input and twenty output nodes in each training pattern. The inputs consist of the log10 value of the resistivity and the difference between the resistivity at sample depths n and n+1. So for an input pattern combining the tools, L045, L105, and L225, we require 120 input PEs and 20 output PEs. The first forty input nodes are from L045, the second forty input nodes are from L105 and the last forty input nodes are from L225. The output nodes still represent twenty points. If an output point corresponds to a boundary, then we output 1.0; otherwise, we output 0.0. To test the generalization of the neural network, 5% Gaussian noise was been added to the training data.

There are 1,043 training patterns in the training set, which covers a resistivity range from 1 to 200 ohm m, and a thickness range from 0.25 m to 6 m. Four different sets of test data were created different combinations of layer thickness and resistivity.

It is worthwhile to emphasize that the desired output value is 1 if the point is on the boundary, 0 otherwise. When the trained neural networks test the new patterns, the output values might not be exactly 1 or 0. Thus, a confidence level must be set. If the output value is larger than 0.5, I consider it to be a boundary. The closer the value is to 1, the more confidence I have in the boundary location.

I compare the performance of different neural networks: MNN, back-propagation (BP) network and RBFN in the NeuralWare Professional II/Plus™ package and the resilient back-propagation and generalized regression neural network (GRNN) in MATLAB®. The results are analyzed according to the average thickness of the layers in the test models.

4.1 MNN, MLP, and RBF Architectures

The MNN has a more complex structure than the other networks. The MNN consists of a gating network and a group of local experts. My best MNN structure required six local experts (Table 15.2). Each training pattern combines the three log responses, which are the fixed length segments of logging curve from L045, L105 and L225. The gating network breaks the problem into six parts, one for each local expert or module, based on the segment's shape and resistivity range. Each local expert learns a particular shape of the segment and resistivity range. The best architecture of the MLP with BP learning and RBFN are shown in Tables 15.3 and 15.4.

Table 15.2
The best architecture of the modular network for the training set

Train. pattern	Gating output	Gating hidden	Local expert hidden	Iter.	Learn. Rule	Trans. function	rms	Learn. rate	Mom.
1043	8	8	6	120,000	Delta-rule	TanH	0.12	0.9	0.4

Table 15.3
The best architecture of the MLP network for the training set

Training patterns	Hidden PEs	Iteration	Learning rule	Transfer function	rms	Learning rate	Mom.
1043	24	120,000	Delta-rule	Tanh	0.167	0.9	0.4

Table 15.4
The best architecture of the RBFN for the training set

Train. patterns	Pattern units	Hidden PEs	Iteration	Learning rule	Transfer function	rms	Learning rate	Mom.
1043	100	10	120,000	Delta-rule	Tanh	0.207	0.9	0.4

4.2 RProp and GRNN Architectures

The MLP with back-propagation learning employed in NeuralWare Professional II ™ uses gradient descent learning. The neural network toolbox in the MATLAB® includes a number of variations, such as resilient back-propagation (RProp), Levenberg-Marquardt and conjugate gradient.

The problem when using steepest decent to adjust the connection weights with sigmoidal transfer functions is that the sigmoidal functions generate a very small slope (gradient) when the input is large, producing small changes in the weights. This makes training very slow. The purpose of RProp is to remove the effects from the small gradients and improve the training speed. Therefore, the magnitude of the derivative has no effect on the weight update in RProp. Instead, the weights are changed based on the sign of the partial derivatives of the cost function.

- If the derivative of the cost function with respect to a weight has the same algebraic sign for two successive iterations, then, increase the update value for the weight.
- If the algebraic sign of the derivative of the cost function with respect to a weight alternates from the previous iteration, then, decrease the update value for the weight.
- If the derivative is zero, then the update value remains same.

The modified algorithm works better than the standard gradient descent algorithm in general and it converges much faster. Table 15.5 lists the best architecture of RProp.

Table 15.5
The best architecture of RProp for the training set

Training patterns	Hidden PEs	Iteration	Transfer function, hidden layer	Transfer function, output layer	rms	Learning rate	Mom.
1043	25	106,300	TanH	Sigmoid	0.09	0.9	0.1

A general regression neural network (GRNN) is in some ways a generalization of a radial basis function network. Just like RBFN, the GRNN has a group of pattern units (centers) to measure the Euclidean distance between input vector and the centers. However, unlike the RBFN and GRNN in NeuralWare™, where input vectors that are close can be clustered together to share a pattern unit, the number of pattern units equals the number of training patterns in GRNN in MATLAB®. That makes GRNN efficient for learning, but susceptible to over-fitting.

Since each pattern unit performs a Gaussian transfer function, the width of the transfer function affects the pattern unit's response area to the input vector. A spread constant, variable name SPREAD, has been applied in GRNN for the pattern units to determine each pattern unit's response area to the input vector. If the SPREAD value is small, the Gaussian transfer function is very steep so that the pattern unit, which is closest to the input vector, generates a much larger output than a more distant pattern unit. If the SPREAD is large, the pattern unit's slope is smoother and several pattern units might correspond to the same input pattern.

The features of GRNN make its architecture very simple; only one parameter, SPREAD, needs to be determined. For the layer-picking problem, the trained network produced the best results when SPREAD was equal to 0.5.

5. ANALYSIS OF RESULTS

5.1. Thin layer model (thickness from 0.5 to 2 m)

The first test set examines the capability of the neural networks to pick thin layer boundaries. Figure 15.4 shows the synthetic responses with 5% Gaussian noise for the thin layer model over a certain depth interval. The layer thicknesses and resistivities are shown in Table 15.6. Thin layer boundaries are always hard to pick from the deep investigation unfocused devices because of the large spacing between the transmitter and receiver. From my previous results (Zhang et al., 1999) using single logging tools, the networks operating on data from L045 and L105 tools could pick most of the boundaries, but the confidence level was relatively low. Since the minimum bed resolution of the L225 tool is 2 m, the L225 network failed to pick the thin layer boundaries. However, when data from all three tools are used together the results improve.

Table 15.6
The layer thickness and resistivities of the thin layer model

Layer number	1	2	3	4	5	6	7	8	9
Resistivity (ohm m)	5	30	80	10	70	1	10	5	30
Thickness (meter)	2	1.5	0.5	1.5	0.5	1.5	2	1.5	

In Figure 15.4, the forward responses for the thin layer make picking the exact layer boundaries difficult. However, the MNN network was able to pick seven out of eight boundaries with high confidence and low noise level. Only one boundary between the 4th layer and 5th layer was missed. The BP network picked five boundaries. However, the boundaries between the 3rd and 4th layer, 4th and 5th layer and 6th and 7th layer were missed. The RBFN had a difficult time picking the boundaries from the thin layer model and only four boundaries were picked.

RProp missed the first boundary, but GRNN missed half of the boundaries. RProp also picked seven of eight boundaries with little noise and high confidence. Therefore, the modified algorithm in RProp increased the convergence speed, as well as improved the generalization capability compared to the BP algorithm.

GRNN had a rapid training rate. The GRNN learned the training set in 20 seconds compared to 3.5 minutes for RProp. However, the algorithm for GRNN produced poor results with only four (1st, 3rd, 5th and 8th) boundaries correctly picked. Many more false boundaries were picked compared with the other networks.

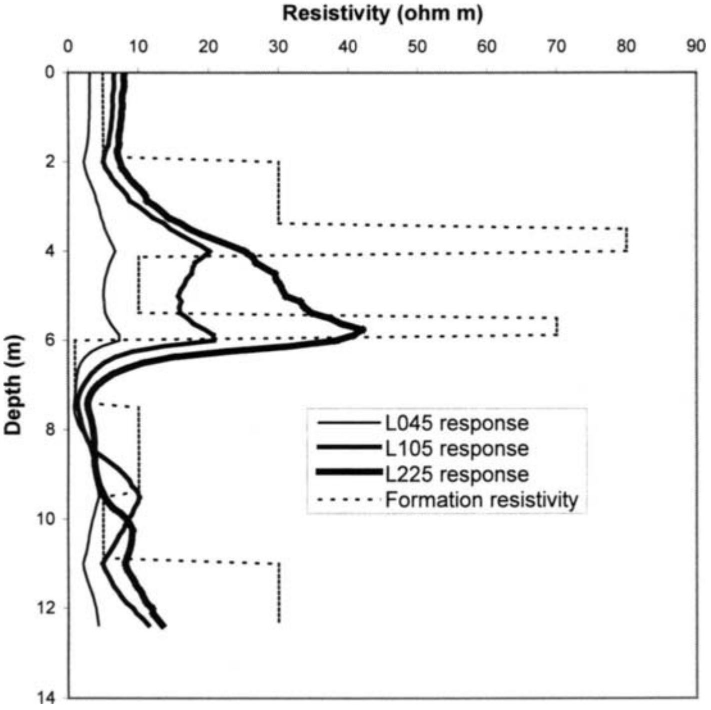


Figure 15.4. Synthetic log responses for the thin layer model with 5% Gaussian noise added. The actual depth locations are irrelevant since the network never receives information on the actual depth of any sample point. All responses are plotted relative to the of the AM spread.

The statistics for picking the layer boundaries from all the trained networks are listed in Table 15.7. In Figures 15.5 to 15.9, the boundary selections are shown graphically.

Table 15.7
Performance of the networks for picking layer boundaries from multiple log responses generated from a thin layer model that has eight boundaries. (Hit means the network picked a true boundary; False alarm (FA) means the network picked a non-existent boundary)

Network	Hits	FA
MNN	7	0
BP	5	1
RBFN	4	0
RProp	7	0
GRNN	4	4

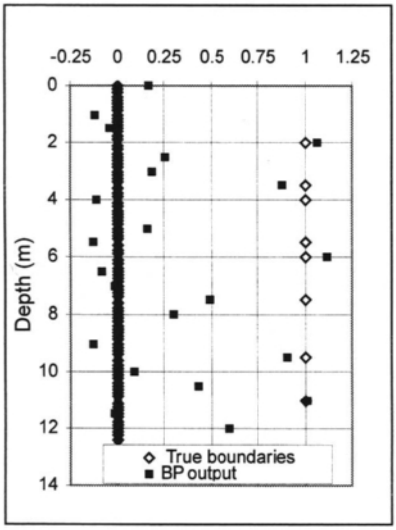


Figure 15.5. Three boundaries are missed and a false boundary is selected at 12 m depth.

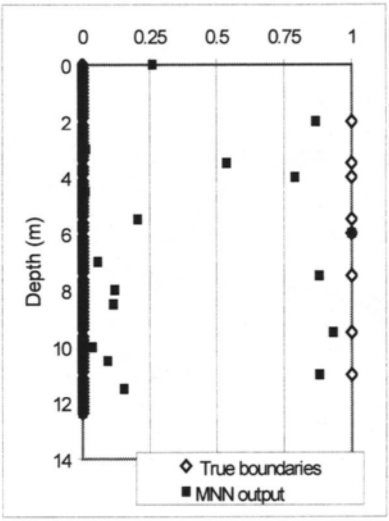


Figure 15.6. The boundary between the 4th and 5th layer is missed.

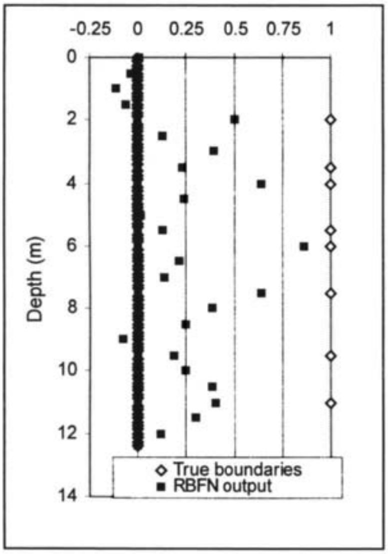


Figure 15.7. Four boundaries are missed.

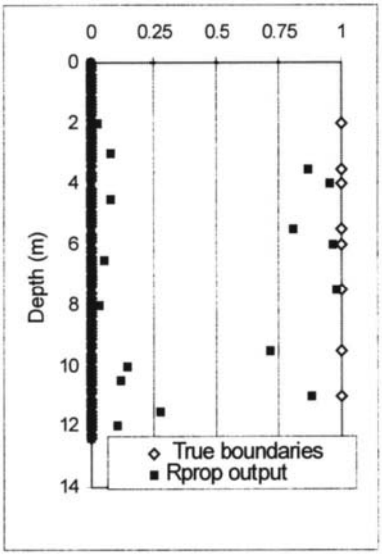


Figure 15.8 Boundaries are picked with high confidence and low noise.

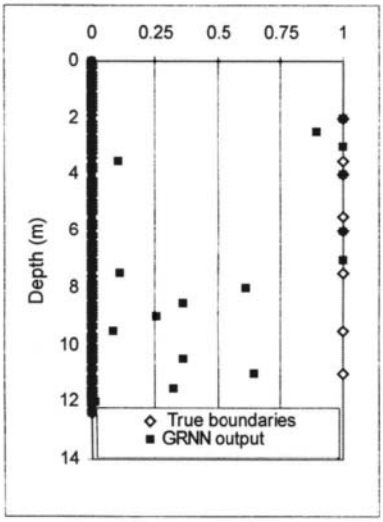


Figure 15.9. GRNN failed to pick most boundaries.

The MNN, RBF, and GRNN networks are designed to cluster or partition the training data into groups with similar characteristics. We can examine the local experts in the MNN to see how the training data were partitioned. Table 15.8 shows the resistivity ranges for each tool and how many patterns each local expert represented.

Table 15.8
Distribution of resistivity in the local experts for the thin-layer model

Local experts	Training patterns	Resistivity (ohm m) L045	Resistivity (ohm m) L105	Resistivity (ohm m) L225
1	121	2—5	4—16	8—20
2	91	5.7—6	20.6—21.8	58—63
3	231	3.4—4.1	9.5—13	18—28
4	255	3.5—6.5	12—22	30-50
5	181	4—6.5	10—22	20—50
6	154	5.85—6.05	22—23	65—70

We can also plot the types of curve segments learned by the local experts (Figure 15.10) to see how they differ much the same way as we plotted seismic waveforms learned by an SOM network in Chapter 10.

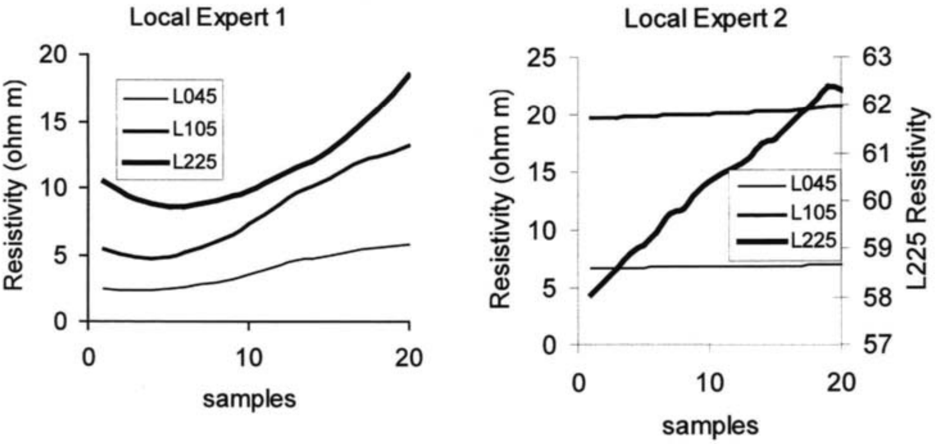


Figure 15.10a. Sample logging curve segments represented by each local expert in the MNN. Some of the L225 data require a separate axis scaling on the right side of the figures.

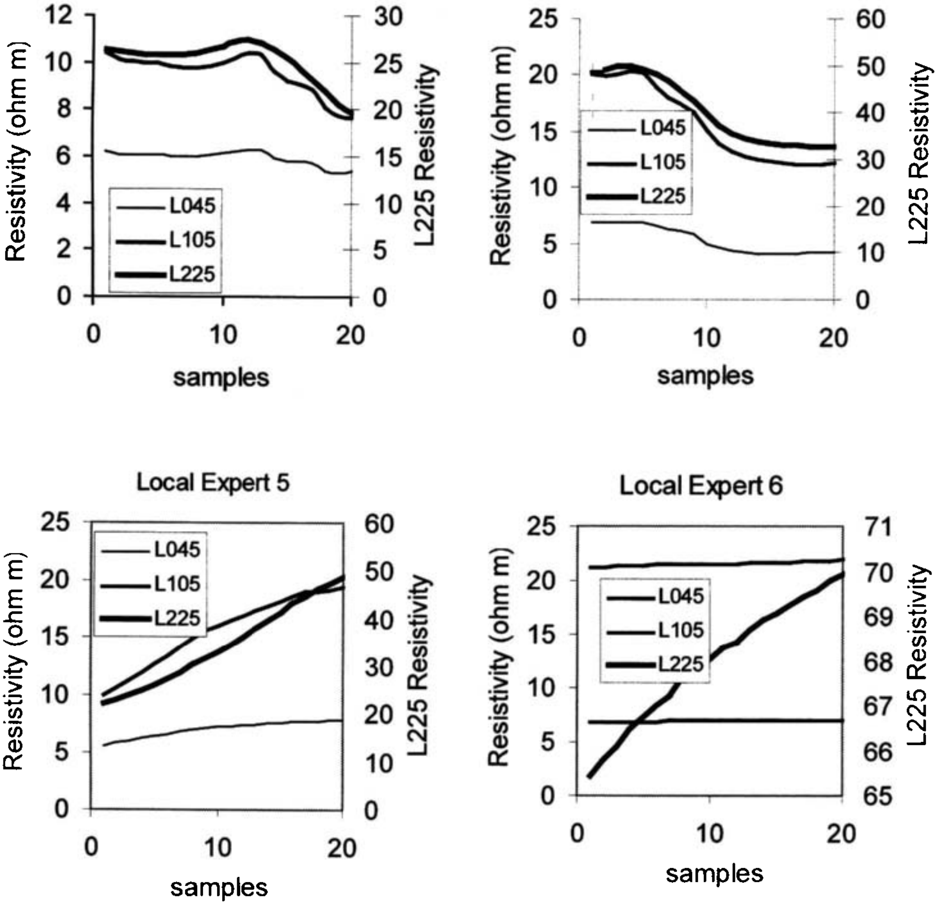


Figure 15.10b. Sample logging curve segments represented by each local expert in the MNN. Note that Experts 2 and 6 differ primarily in the resistivity magnitude.

5.2. Medium-thickness layer model (thickness from 1.5 to 4 m)

I next tested the capability of the networks to pick layer boundaries from a medium-thickness layer model, all the thicknesses in this case are from 1.5 to 4 m. Figure 15.11 shows the synthetic responses with 5% Gaussian noise for the normal thickness layer model over a certain depth interval. The layer thicknesses and resistivities are shown in Table 15.9.

Table 15.9
Resistivities and thickness in the medium-thickness layer model

Layer number	1	2	3	4	5	6	7	8	9	10	11	12	13
Resistivity (ohm m)	45	15	75	50	100	55	35	1	20	5	10	1	15
Thickness (meter)	2	2.5	3.5	1.5	4	2.5	3	2	3	4	2	3	

The statistics for picking the layer boundaries from the networks are listed in Table 15.10. The boundary selections are shown in Figures 15.12 to 15.16.

For the medium-thickness model, the BP network picked all the layer boundaries with high confidence and less noise. The MNN missed one boundary between the 4th and 5th layers. However, the output value for this boundary was 0.493, only slightly less than the threshold limit of 0.5 required for picking the boundary. All the output values for the picked boundaries were higher than 0.75 for the MNN. The RBFN still could not pick layer boundaries very well and the output values for the picked boundaries have lower confidence levels.

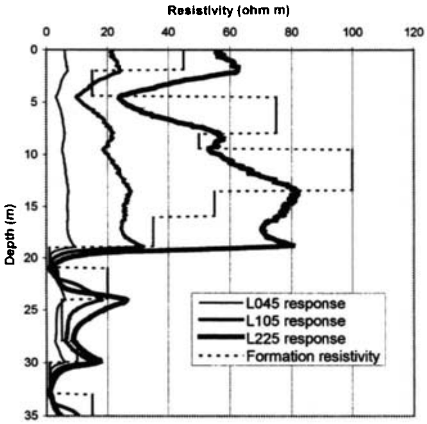


Figure 15.11. Synthetic log responses for the medium-thickness layer model with 5% Gaussian noise added. The actual depth points on the curves are irrelevant since the network never uses the depth for any sampling point.

RProp missed the 3rd boundary. Although the BP network in NeuralWare Professional II™ picked all the boundaries, the outputs of RProp definitely have less noise.

The GRNN in MATLAB® performed better than the RBF network in NeuralWare Professional II™, which picked only five boundaries. Although GRNN picked only eight of the 12 boundaries, all the output values for the picked boundaries were more than 0.9. For this data set the GRNN produced the most consistent and highest confidence output values for the layer boundaries.

Table 15.9
Performance statistics for the networks for a medium-thickness layer model that has twelve boundaries. (Hit means the network picked the true boundary; False alarm (FA) means the network picked a nonexistent boundary).

Network	Hits	FA
MNN	11	0
BP	12	0
RBFN	5	0
RProp	11	0
GRNN	8	5

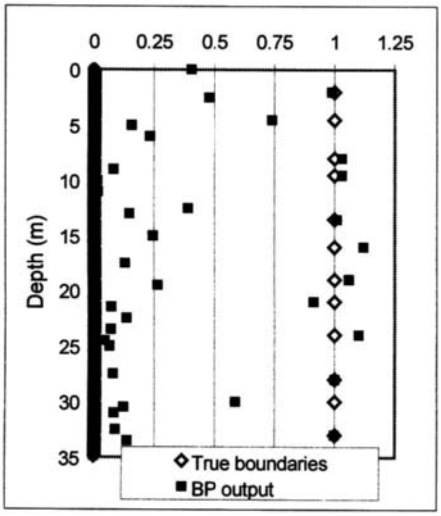


Figure 15.12. BP output boundaries for the medium-thickness model compared to true boundaries. All the boundaries are correctly picked.

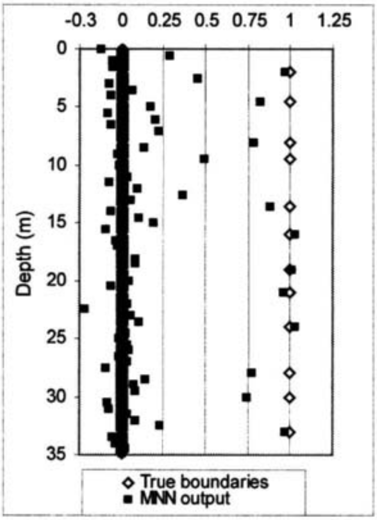


Figure 15.13. For the MNN, the boundary between the 4th and 5th layers is missed but the output value for this boundary is 0.493, just below the threshold of 0.5 for correct classification.

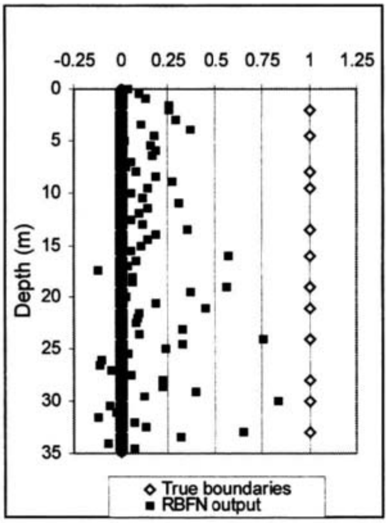


Figure 15.14. The RBF network missed 7 boundaries.

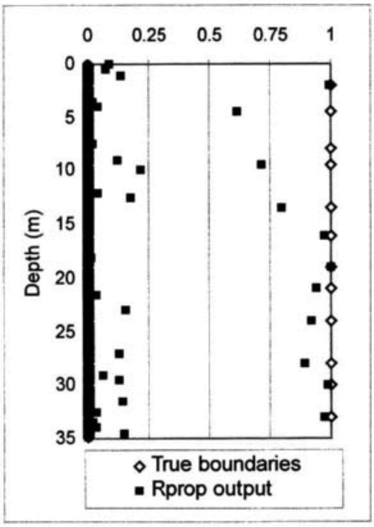


Figure 15.15. Most boundaries are picked with high confidence.

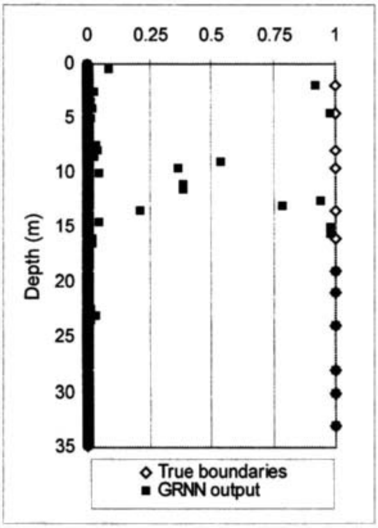


Figure 15.16. Most GRNN output boundaries are picked with high confidence.

5.3 Thick layer model (thickness from 6 to 16 m)

The third test was to probe the capability of the neural networks for picking layer boundaries from a thick layer model. All the thickness values in the case are from 6 to 16m. Note, however, that the training set did not include layers thicker than 6 m. Figure 15.17 shows the synthetic responses with 5% Gaussian noise for the thick layer model over a certain depth interval. The layer thicknesses and resistivities are shown in Table 15.10.

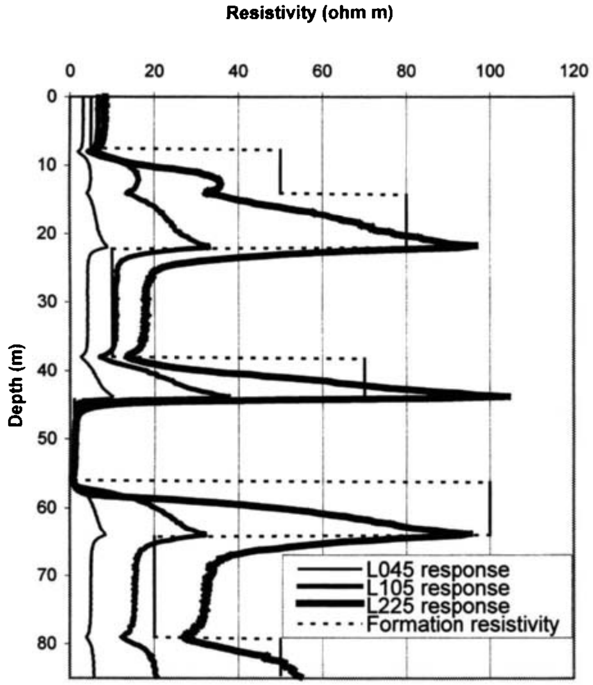


Figure 15.17. Synthetic log responses for the thick layer model with 5% Gaussian noise added. The actual depth points for the logging curves are irrelevant to the network interpretation.

Table 15.10
The resistivities and thickness in the thick-layer model

Layer number	1	2	3	4	5	6	7	8	9
Resistivity (ohm m)	5	50	80	10	70	1	100	20	50
Thickness (meter)	8	6	8	16	6	12	8	15	

The network statistics for picking the layer are listed in Table 15.11. Graphical results are shown in Figures 15.18 to 15.22.

In general, thick layer boundaries are easier to pick than thin layer boundaries. The MNN picked all the boundaries successfully with high confidence and less noise. The BP network missed the second boundary, which is between the 2nd and 3rd layer. Instead, it picked another boundary that was 1 m shallower than the true boundary. Another false boundary was picked at a depth of 26 m. The RBF network picked three boundaries correctly and the confidence level was relatively low.

The RProp network also missed the second boundary. Based on Figure 15.17, there is little evidence of the boundary in the forward responses. Compared to the BP network in NeuralWare Professional II ™ the RProp network performed better with less noise and higher confidence. All the output values for the picked layer boundaries were higher than 0.9.

GRNN picked the 1st, 4th, 6th and 7th boundaries. However, there were seven false boundaries selected. The RBF network in Professional II™ picked three boundaries correctly but had one false boundary. The GRNN tended to pick more false boundaries than the RBF network because of the narrow width of Gaussian transfer function, which makes the network respond to a target vector very close to the nearest pattern unit. The algorithm of the RBF network in Professional II™ avoided picking many false boundaries because the width of the transfer function was set to the root mean square distance of the given pattern unit to the P nearest neighbor pattern units.

Table 15.11
Performance of the networks for picking layer boundaries from multiple log responses generated from a thick layer model that has eight boundaries. (Hit means the network picked a true boundary; False alarm (FA) means the network picked a non-existent boundary).

Network	Hits	FA
MNN	8	0
BP	7	2
RBFN	3	1
RProp	7	0
GRNN	4	7

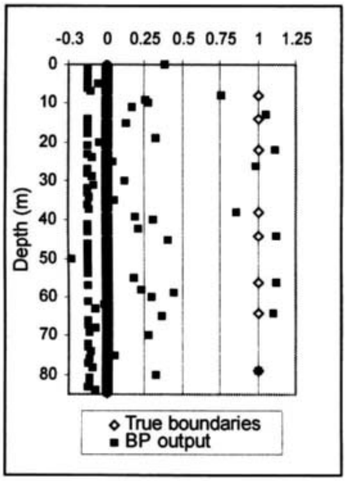


Figure 15.18. BP output boundaries for the thick layer model compared to true boundaries. The boundary between the 2nd and 3rd layer was missed but a boundary 1 m shallower than the true boundary was selected. Another false boundary is picked at 26 m.

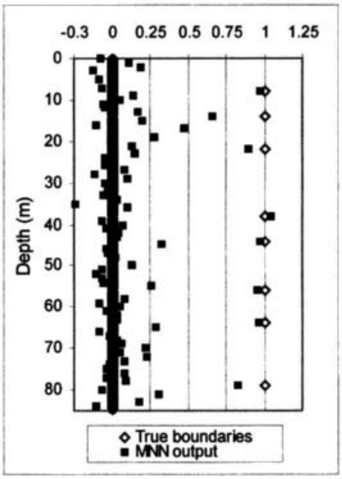


Figure 15.19. MNN output for the thick layer model compared to the true boundaries. All the boundaries are correctly picked.

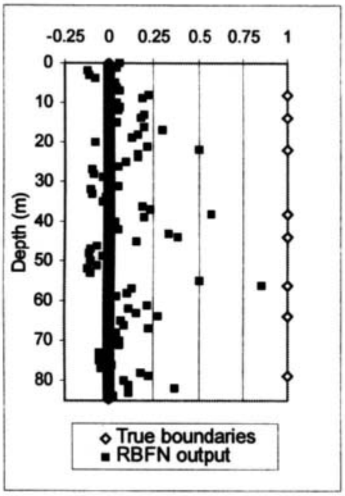


Figure 15.20. For the RBF, five boundaries were missed. Only the 3rd, 4th, and 6th boundaries were picked correctly.

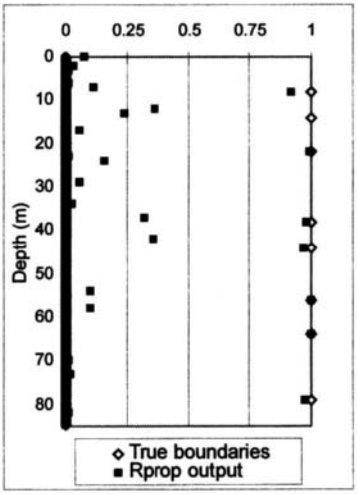


Figure 15.21. For the RProp all the boundaries were picked with high confidence and little noise except the boundary between the 2nd and 3rd layer.

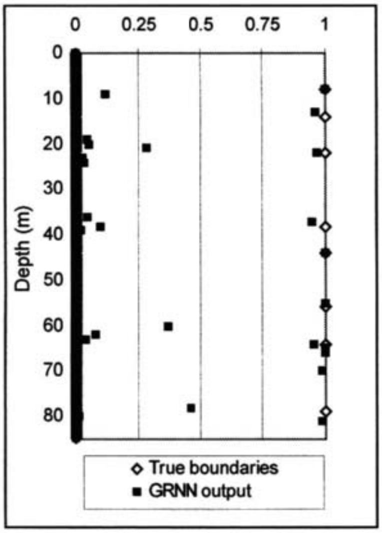


Figure 15.22. GRNN output for the thick layer model compared to the true boundaries. Seven false boundaries were picked.

5.4 Testing the sensitivity to resistivity

The range of resistivity data in the training files is from 1 to 200 ohm m. To determine how well the networks can extrapolate to resistivities outside this range, a new test set is generated. The resistivities in this new test set ranges from 0.1 to 300 ohm m. Figure 15.23 shows the synthetic responses with 5% Gaussian noise for the model over a certain depth interval. The layer thicknesses and resistivities are showed in Table 15.12.

Table 15.12
The resistivity and thickness for the model with extended resistivity range

Layer number	1	2	3	4	5	6	7	8	9	10	11	12	13
Resistivity (ohm m)	80	150	120	300	100	50	100	10	.5	30	.1	20	1
Thickness (meter)	5	3	5	6	8	6	9	6	8	6	4	6	

The statistics for picking the layer boundaries are listed in Table 15.13. Figures 15.24 to 15.28, show the layer boundary selections.

The first boundary in the model in Figure 15.23 is barely detectable and all the networks missed this boundary except the RProp network. Other than the first boundary, all the boundaries are picked correctly by MNN and BP network with high confidence level (more than 0.7). The RBFN picked five boundaries correctly. The GRNN picked seven boundaries correctly but also had nine false alarms.

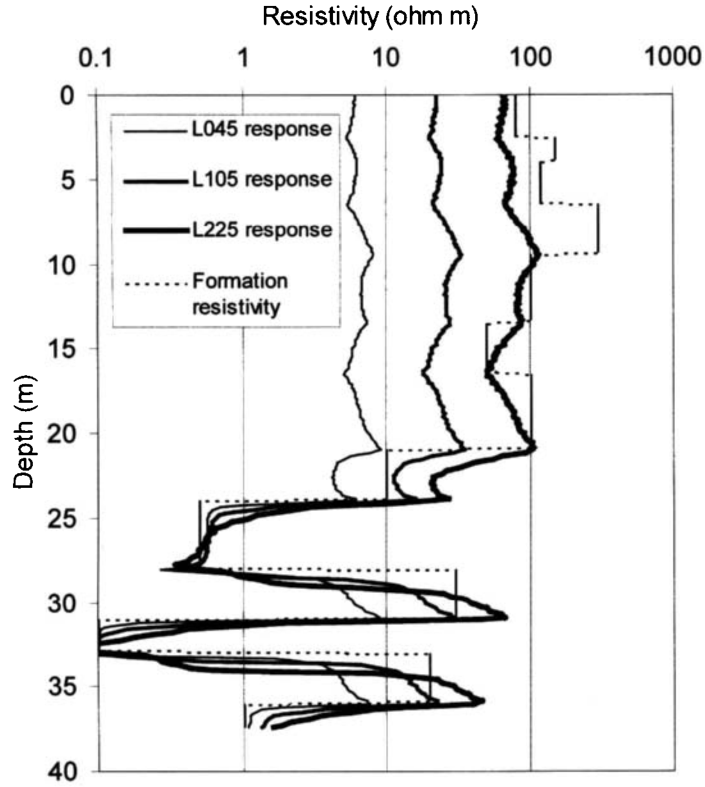


Figure 15.23. Model for testing the range of resistivity.

Table 15.13
Performance of the networks for picking layer boundaries from multiple log responses generated from a model with expanded resistivity range that has 12 boundaries. (Hit = network picks a true boundary; False alarm (FA) = network picks a non-existent boundary).

Network	Hits	FA
MNN	11	0
BP	11	0
RBFN	5	1
RProp	12	0
GRNN	7	9

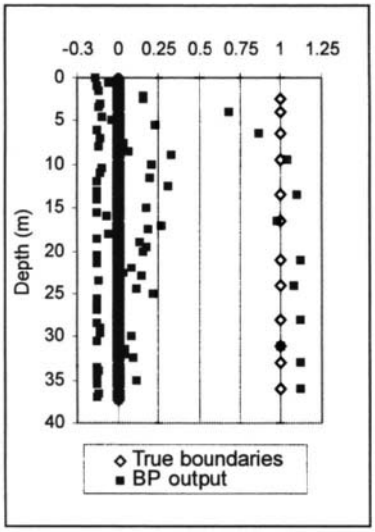


Figure 15.24. BP output for the resistivity model compared to the true boundaries. The first boundary was missed.

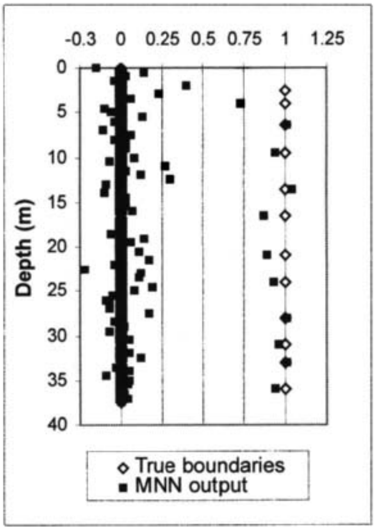


Figure 15.25. MNN output for the resistivity model compared to the true boundaries. The first boundary was missed.

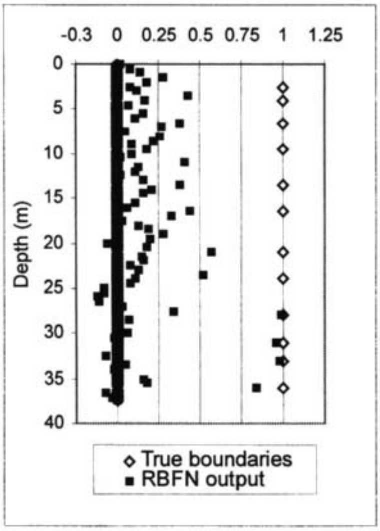


Figure 15.26. The RBF missed seven boundaries.

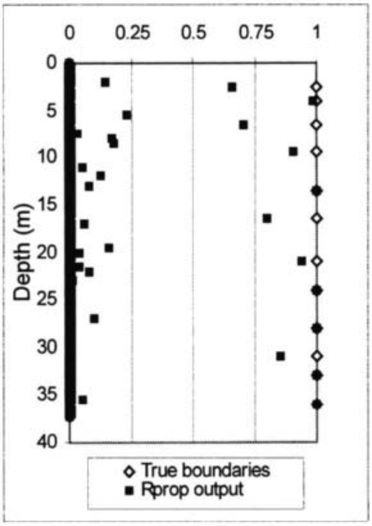


Figure 15.27. RProp picked all boundaries with high confidence.

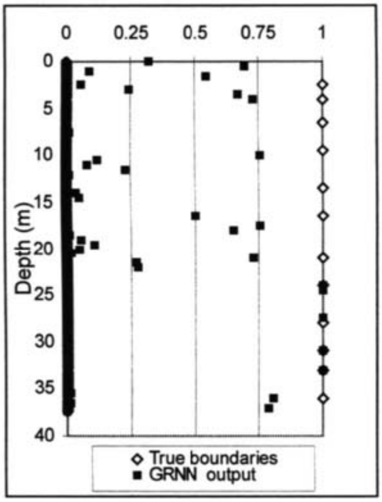


Figure 15.28. GRNN output for the resistivity model compared to the true boundaries. Nine false boundaries were picked.

6. CONCLUSIONS

From the above results, it is clear that the MNN, RProp, and BP networks were successful at picking layer boundaries in data from unfocused logging tools. The modified algorithm in RProp produces layer picks with high confidence and low noise. It is comparable in accuracy with the MNN in Professional II™.

The gating network in the MNN partitioned the training set into several parts based on the shape and values of the training patterns. Thus, each local expert could focus on learning a smaller data set. While the RBF network and GRNN also cluster the training data, the method used by the MNN proved more effective.

The RBF network has a group of pattern units (centers) that measure the Euclidean distance between the input vector and the centers. The input pattern is assigned to the center that has the minimum distance with the input pattern itself. A Gaussian transfer function is performed. The functionality of the pattern units is like a self-organizing phase to organize the input patterns around a different center. The difference between the self-organizing phase in a RBF compared to the MNN is that the clustering phase in the RBF is based on a distance between a prototype and actual pattern whereas in the MNN it is error driven. Hence, for this layer-picking problem, the RBF network does not perform as well because each training pattern consists of three segments of log responses from the three unfocused tools, and the resistivity range in each training pattern is quite different for the same model. For example, the L225 tool has a higher apparent resistivity and the L045 tool has a lower apparent resistivity for the same model. Thus, it is difficult for the RBF network to distribute these training patterns to the prototype centers.

The GRNN picked boundaries with high confidence but tended to pick too many false boundaries. The small SPREAD value gave the Gaussian transfer function a steep slope and each pattern unit responded to a single input vector. The accuracy on the test data was highly dependent on the similarity between the test vector and the pattern unit. Therefore, more training patterns would be required for accurate test results.

The advantages for using data from all three tools simultaneously are:

1. The shallow unfocused tool, L045, has better layer determination for thin layer boundaries; the deep unfocused tool, L225, has poor minimum bed resolution (2 m). However, L225 has a very strong response for thick layer boundaries.
2. Using multiple logs produces higher confidence levels for picking the layer boundaries. Most layer boundaries produced output values greater than 0.7.
3. The noise level is reduced so fewer false alarms are likely to occur.

REFERENCES

Beard, D., Zhou, Q., and Bigelow, E., 1996, Practical applications of a new multichannel and fully digital spectrum induction system: Presented at the SPE Annual Technical Conference and Exhibition.

Chakravarthy, S., Chunduru, R., Mezzatesta, A., and Fanini, O., 1999, Detection of Layer Boundaries from Array Induction Tool Responses using Neural Networks: Society of Exploration Geophysicists 69th Annual International Meeting and Exposition.

Ellis, D., 1987, Well Logging for Earth Scientists: Elsevier Science Publishing.

Haykin, S., 1994, Neural Networks: A Comprehensive Foundation: Macmillan.

Rider, M., 1996, The Geological Interpretation of Well Logs, 2nd Edition: Caithness, Whittles Publishing.

Roy, A., 1981, Focused resistivity logs, *in* Fithch, A., Ed., Developments in Geophysical Exploration Methods: Applied Science Publishers, Chapter 30.

Whitman, W., 1995, Interpretation of unfocused resistivity logs: The Log Analyst, January-February, 35-39.

Whitman, W., Towle, G., and Kim, J., 1989, Inversion of normal and lateral well logs with borehole compensation: The Log Analyst, January-February, 1-11.

Whitman, W., Schon, J., Towle, G., and Kim, J., 1990, An automatic inversion of normal resistivity logs: The Log Analyst, January-February, 10-19.

Yang, F., and Ward, S., 1984, Inversion of borehole normal resistivity logs: Geophysics, **49**, 1541-1548.

Zhang, L., Poulton, M., Mezzatesta, A., 1999, Neural network based layer picking for unfocused resistivity log parameterization: SEG Expanded Abstracts, 69th Annual International Meeting and Exposition.

This Page Intentionally Left Blank

Chapter 16
A Neural Network Interpretation System For Near-Surface Geophysics Electromagnetic Ellipticity Soundings

Ralf A. Birken

1. INTRODUCTION

A radial basis function neural network interpretation system has been developed to estimate resistivities from electromagnetic ellipticity data in a frequency range from 1 kHz to 1 MHz for engineering and environmental geophysical applications. The interpretation system contains neural networks for half-space and layered-earth interpretations. The networks were tested on field data collected over an abandoned underground coal mine in Wyoming. The goal of this investigation was to provide subsurface information about areas of subsidence, which were caused by an underground coal mine fire.[*]

The frequency-domain electromagnetic imaging system used in this study was designed for shallow environmental and engineering problems with the goals of high accuracy data, rapid data collection, and in-field interpretation (Sternberg and Poulton, 1994). The system recorded soundings between 1 kHz and 1 MHz typically at 8, 16, or 32 meter coil separations but other separations could also be used. The transmitter was a vertical magnetic dipole and used a sinusoidal signal supplied from an arbitrary waveform generator via a fiber optic cable. The receiver was a tuned 3-axis coil. The acquired magnetic field data were mathematically rotated to the principal planes, signal-averaged, filtered, and stored on a field computer before being transferred to the interpretation computer via a radio-frequency telemetry unit. The interpretation computer was located in a remote recording truck and could display the data for interpretation in near real-time in the field using neural networks. The transmitter and receiver equipment were mounted on 6-wheel drive all-terrain vehicles. Eleven frequencies were transmitted in binary steps over the frequency range.

The electromagnetic ellipticity was calculated based on three components of the magnetic field (Bak et al., 1993; Thomas, 1996; Birken, 1997). Using the rotated complex magnetic field vector $\vec{H}' = H_1' \cdot \vec{e}_1 + H_2' \cdot \vec{e}_2 + H_3' \cdot \vec{e}_3$ the *3D-ellipticity* is calculated using equation (1), where $\vec{e}_j$ for $(j = 1,2,3)$ are unit vectors in Cartesian coordinates.

---

[*] The field study was funded by the U.S. Bureau of Mines, Abandoned Mine Land Program, contract #1432-J0220004.

$$3D - Ellipticity = (-1)\cdot\frac{|Minor|}{|Major|} = (-1)\cdot\frac{|\mathrm{Im}(\vec{H}')|}{|\mathrm{Re}(\vec{H}')|} = (-1)\cdot\frac{\sqrt{H_{1i}'^{2} + H_{2i}'^{2} + H_{3i}'^{2}}}{\sqrt{H_{1r}'^{2} + H_{2r}'^{2} + H_{3r}'^{2}}}. \tag{16.1}$$

The trained neural networks were integrated in a data visualization shell. The data visualization shell provided the user interface to the neural networks, graphs of sounding curves, 1D forward modeling program, images of the data, and interpreted sections. The only interaction the user had with the trained neural networks was the selection of the trained networks to use for the interpretation through the visualization shell.

The Ellipticity Data Interpretation and visualization System (EDIS) was developed based on the Interactive Data Language 3.6.1 (IDL) computing environment for the Windows operating system on a personal computer. EDIS is a Graphical User Interface (GUI) that visualizes ellipticity data and their interpretations and manages over 100 trained interpretation neural networks (Birken, 1997).

Display capabilities in EDIS are for sounding curves, interpreted resistivity and relative dielectric constant sections, and raw ellipticity sections. The user may select up to twelve sounding curves to display at one time. The difference between the last two selected curves is automatically displayed on a graph below the sounding curves. Interpreted data are displayed in 2D pseudo-depth-sections that show the color-coded resistivities or relative dielectric constants. The y-axis of the sections indicates the depths of the interpreted layers. Several sections can be displayed at one time for direct comparison, for example, for different offsets or lines. Raw ellipticity line data can be displayed versus frequency number or depth of investigation.

The user selects all the networks through which the data should be routed. Each network interpretation is passed to a 1D forward modeling code so the ellipticity curves can be compared to the measured data. The fit of each interpreted sounding to the field data is calculated as the mean-squared error for the number of frequencies in each sounding. The user decides which network gives the best fit and picks that network for the interpretation. The network is re-run for the sounding, and the interpretation is plotted in a 2D section. After deciding a particular neural network for the interpretation of a specific station, the neural network results are stored on the hard disk and can be used to interactively construct a resistivity, relative dielectric constant or ellipticity section. In addition, 1D forward modeling and inversion capabilities limited to three layers are also included.

The neural networks implemented serve two major functions: interpretation of half-space and layered-earth models. The half-space networks consist of one network that uses nine or ten frequencies to estimate a half-space resistivity and nine networks that use the ellipticity pairs for adjacent frequency to estimate a half-space resistivity for each pair (Figure 16.1 in Section 3). We will refer to the first network as a half-space network and the other eight or nine networks as piecewise-half-space resistivity networks. The main advantage of the piecewise half-space networks is the ability to fit segments of the sounding curve and to more easily deal with bad or missing data. The layered-earth networks estimate the resistivities and thickness for two or three layers. We will not discuss the layered-earth networks in this chapter.

A typical system dependent dataset contains 11 ellipticity values at 11 frequencies, in which in many cases the highest frequency (1 MHz) is noisy. Therefore, we consider only 10 ellipticity values as input to our neural networks for our study.

2. FUNCTION APPROXIMATION

The problem at hand is a function approximation problem. The function describes the physical relationship between the Earth material property resistivity and the measured geophysical quantity *3D-ellipticity* (Eq. (16.1)). In this section I provide a brief overview of a few function approximation techniques and how they compare or relate to a radial basis function neural network.

2.1. Background

Learning an input-output mapping from a set of examples can be regarded as synthesizing an approximation of a multidimensional function (that is, solving the problem of hypersurface reconstruction from sparse data points) (Poggio and Girosi, 1990a). Poggio and Girosi point out that this form of learning is closely related to classical approximation techniques, such as generalized splines and regularization theory. In this context Poggio and Girosi (1990b) describe *learning* simply as collecting examples, i.e. the input corresponds to a given output, which together form a look-up-table. *Generalization* is described as estimating the input where there are no examples. This requires approximation of the surface between the example data points most commonly under the assumption that the output varies smoothly (i.e. small changes in input parameters cause a correspondingly small change in the output parameters) and therefore can be called *hypersurface reconstruction*. Bishop (1995) points out that the best generalization to new data is obtained when the mapping represents the underlying systematic aspects of the data rather then capturing the specific details (i.e. noise contribution). Note that generalization is not possible if the underlying function is random, e.g. the mapping of people's names to their phone numbers (Poggio and Girosi, 1990b). And the best generalization is determined by the trade-off between two competing properties, which Geman et al. (1992) investigate by decomposing the error into *bias* and *variance* components (see Chapter 3).

Poggio and Girosi (1990b) point out that techniques that exploit smoothness constraints in approximation problems are well known under the term of standard regularization. A standard technique in regularization theory (Tikhonov and Arsenin, 1977) is to solve the problem by minimizing a cost functional containing two terms

$$H[f] = \sum_i (z_i - d_i)^2 + \lambda \|Pf\|^2, \tag{16.2}$$

where the first term measures the distance between the data z_i and the desired solution d_i and the second term measures the cost associated with the deviation from smoothness. The index i represents all known data points and $\|Pf\|$ represents the regularization term and depends on the mapping function and is designed to penalize mappings that are not smooth. λ is the regularization parameter controlling the extent to which $\|Pf\|$ influences the form of the solution and hence the complexity of the model (Bishop, 1995), i.e. it influences the generalization and the trade-off between bias and variance.

Functions f that minimize the functional in Eq. (16.2) can be generalized splines (Poggio and Girosi, 1990a,b). To close the loop to the radial basis function neural networks described next, Poggio and Girosi (1990a,b) show that they are closely related to regularization networks, which are equivalent to generalized splines.

2.2. Radial basis function neural network

Radial basis function (RBF) neural networks are a class of feed forward neural network implementations that are not only used for classification problems, but also for function approximation, noisy interpolation and regularization. RBF methods have their origins in work by Powell (1987), in which he shows that RBFs are a highly promising approach to multivariable interpolation given irregularly positioned data points. This problem can be formulated as finding a mapping function f that operates from a n-dimensional input or data space $\Re^n$ to a one-dimensional output or target space $\Re$, which is constrained by the interpolation condition,

$$f(\vec{x}_i) = y_i \quad \forall i = 1,2,\ldots,P, \tag{16.3}$$

where each of the P known data points consist of an input vector $\vec{x}_i$ and a corresponding real value y_i. The system of functions used for this interpolation is chosen to be from the set of RBFs b_i, which depend on the selection of the known data points $\vec{x}_i$. The RBFs $b_i(\|\vec{x}-\vec{x}_i\|) \quad \forall i = 1,2,\ldots,P$ are continuous non-linear functions, where the i-th RBF b_i depends on the distance between any data point $\vec{x}$ and the i-th known data point $\vec{x}_i$, typically the Euclidean norm of $\Re^n$. Therefore, the mapping function can be approximated as a linear combination of the RBFs b_i with the unknown coefficients w_i,

$$f(\vec{x}) = \sum_{i=1}^{P} w_i \cdot b_i(\|\vec{x}-\vec{x}_i\|). \tag{16.4}$$

Inserting the interpolation condition (16.2) in the mapping function (16.3) results in a system of linear equations for the w_i

$$\sum_{i=1}^{P} w_i \cdot b_i(\|\vec{x}_j - \vec{x}_i\|) = y_j \quad \forall j = 1,2,\ldots,P, \tag{16.5}$$

which can be rewritten in matrix notation as,

$$B\vec{w} = \vec{y} \quad with\ \vec{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_P \end{bmatrix}, \vec{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_P \end{bmatrix}\ and\ B = \begin{bmatrix} b_1(\|\vec{x}_1 - \vec{x}_1\|) & \cdots & b_P(\|\vec{x}_1 - \vec{x}_P\|) \\ \vdots & \ddots & \vdots \\ b_1(\|\vec{x}_P - \vec{x}_1\|) & \cdots & b_P(\|\vec{x}_P - \vec{x}_P\|) \end{bmatrix}. \tag{16.6}$$

Equation (16.5) can be solved by inverting the matrix B, assuming its inverse matrix B^{-1} exists

$$\vec{w} = B^{-1}\vec{y}. \tag{16.7}$$

Poggio and Girosi (1989) point out several interesting mathematical characteristics of the RBFs b_i and the matrix B. They demonstrate that the matrix B is non-singular for a large class of functions b_i (assuming that the $\vec{x}_i$ are distinct data points), following findings by Micchelli (1986). Poggio and Girosi (1989) also showed that for RBF neural networks of the type described above the best approximation property exists and is unique. This does not hold for multi-layer Perceptrons of the type used in back propagation networks and also not for the generalized RBF neural network (Girosi and Poggio, 1990), which is described below. Light (1992) showed that B is positive definite, as summarized in Haykin (1994). So, the solution of equation (16.5) provides the coefficients or weight values w_i of equation (16.3), which makes the interpolation function $f(\vec{x})$ a continuous differentiable function containing each of the data points $\vec{x}_i$.

At this point it is appropriate to generalize the formulation to a mapping function f that operates from a n-dimensional input space $\Re^n$ to a m-dimensional output space $\Re^m$, which is equivalent to a mapping of m functions $f_k \quad \forall k = 1,2,\dots,m$ from $\Re^n \to \Re$. So the resulting interpolation condition can be written as

$$f_k(\vec{x}_i) = y_i^k \quad \forall i = 1,2,\dots,P \quad \forall k = 1,2,\dots,m, \tag{16.7}$$

where each of the P known data points consist of an input vector $\vec{x}_i$ and a corresponding real output vector $\vec{y}_i$ with components $y_i^k \quad \forall k = 1,2,\dots,m$. The f_k are obtained as in the single-output case (Eq. (16.4)) by linear superposition of the P RBFs h_i

$$\sum_{i=1}^{P} w_i^k \cdot b_i(\|\vec{x}_j - \vec{x}_i\|) = y_i^k \quad \forall i,j = 1,2,\dots,P \quad \forall k = 1,2,\dots,m\ , \tag{16.8}$$

where the weight values w_i^k are determined by

$$w_i^k = \sum_{n'=1}^{P} \left(B^{-1}\right)_{in'} y_{n'}^k . \tag{16.9}$$

Note that for an numerical evaluation of equation (16.9) B^{-1} only needs to be calculated once. Haykin (1994) and Zell (1994) point out that for all practical purposes the inverse matrix B^{-1} will not be determined by inverting B, rather through some efficient, numerical stable algorithm that solves large systems of linear equations such as given by equation (16.6). One solution can be found in the regularization theory, in which in general a small perturbation term would be added to the matrix $B + \lambda \underline{\underline{I}}$.

So far I have discussed that an interpolation function $f(\vec{x})$ using RBFs b_j can be found that honors the interpolation condition requiring that all given data points x_j are part of the solution. This can lead to several unwanted effects as pointed out e.g. by Zell (1994) and Bishop (1995). One being strong oscillations between the known data points, which is a well known effect from the interpolation of higher order polynomials, introduced by the interpolation condition forcing $f(\vec{x})$ to pass exactly through each data point. In many cases an exact interpolation is not desired, because the known input data have noise associated with them, a smoother solution would be more appropriate. The size of the system of linear equations is proportional to the number of known points $\vec{x}_j$, which is also an unwanted effect.

These problems lead to the implementation of a number of modifications to the exact interpolation formula (Eq. (16.8)), the most important being a fixed size for the system of linear equations,

$$y^k = f_k(\vec{x}) = \sum_{j=1}^{M} w_j^k \cdot b_j(\|\vec{x} - \vec{\mu}_j\|), \tag{16.10}$$

where M is the number of RBFs, which is typically much less then the number P of known data points. The vectors $\vec{\mu}_j$ are the centers of the RBFs b_j and are no longer constrained to the known input vectors $\vec{x}_j$. The above statements hold for a whole group of RBFs b_j (Poggio and Girosi, 1989) from which the most common used in a neural network is a Gaussian

$$G_j(\vec{x}) = \exp\left(\frac{\|\vec{x} - \vec{\mu}_j\|^2}{2 \cdot \sigma_j^{\,2}}\right) \quad for\ \sigma_j > 0 \quad \forall j = 1,2,\ldots,M. \tag{16.11}$$

Assuming that a Gaussian G_j (Eq. (16.10)) is used in a generalized radial basis function (GRBF) neural network (Broomhead and Lowe, 1988; Moody and Darken, 1989; Poggio and Girosi, 1989; Girosi and Poggio, 1990; Musavi et al., 1992, Haykin, 1994; Zell, 1994; Bishop, 1995), then not only the centers $\vec{\mu}_j$ are calculated during the network training, but also the widths σ_j of each G_j. Both are calculated during the initial unsupervised training phase as described later.

The neural network implementation of the RBF approximation discussed above consists of one hidden network layer in which each processing element evaluates a RBF on the incoming signal and an output layer that computes a weighted linear sum using RBFs as transfer functions. The M radially symmetric RBFs actually used in this study are normalized Gaussian functions, another specific example of RBFs (Hertz et al., 1991)

$$\hat{G}_j(\vec{x}) = \frac{\exp[-(\vec{x}-\vec{\mu}_j)^2/2\sigma_j^2]}{\sum_{k=1}^{M}\exp[-(\vec{x}-\vec{\mu}_k)^2/2\sigma_k^2]}, \tag{16.12}$$

which have maximum response when the input vector $\vec{x}_i$ is close to their centers $\vec{\mu}_j$ and decrease monotonically as the Euclidean distance from the center increases. Each of the $\hat{G}_j$ (note that there are fewer RBFs then known data points) responds best to a selected set of the known input vectors. If a vector activates more than one $\hat{G}_j$ then the network response becomes a weighted average of the two Gaussians. Therefore the RBF neural network makes a sensible smooth fit to the desired non-linear function described by the known input vectors $\vec{x}$.

The *hybrid* RBF neural network used in this study is a combination of a standard RBF neural network as just described, which is trained unsupervised, and a back-propagation neural network. The latter uses the output of the RBF neural network as input to a subsequent supervised learning phase. The first unsupervised training phase consists of finding centers, widths and weights connecting hidden nodes to output nodes. A K-means clustering algorithm (Spath, 1980; Darken and Moody, 1990) is used to find the centers $\vec{\mu}_j$ of the $\hat{G}_j$. A nearest neighbor approach is used to find the widths σ_j of the $\hat{G}_j$. The centers $\vec{\mu}_j$ are initialized randomly and then the distance from each known input training pattern to each center is calculated. The closest center to each training pattern $\vec{x}$ is modified as

$$\vec{\mu}_j^{(new)} = \vec{\mu}_j^{(old)} + \eta\cdot(\vec{x}_i - \vec{\mu}_j^{(old)}) \ , \tag{16.13}$$

where η is the step-size. The widths σ_j of the $\hat{G}_j$ are found by setting them to the root-mean-square-distances of the cluster centers to the A nearest neighbor cluster centers

$$\sigma_j = \sqrt{\frac{1}{A}\sum_{a=1}^{A}\left|\vec{\mu}_j - \vec{\mu}_a\right|^2} \tag{16.14}$$

After the centers and widths of all RBFs $\hat{G}_j$ have been found, it is time to determine the w_j^k according to equation (16.10). There are several ways of optimizing the w_j^k. One of them is to minimize a suitable error function and use the pseudo-inverse solution as described by Bishop (1995). In practice single-value decomposition is used to avoid possible problems with ill-conditioned matrices (Bishop, 1995). Now the second supervised training phase may begin. This learning phase uses an additional hidden layer in the network in which case training proceeds as in standard back-propagation with the input to the second hidden layer being represented by the output of the RBF neural network.

3. NEURAL NETWORK TRAINING

Nine different piecewise half-space neural networks were trained for each transmitter-receiver (Tx-Rx) separation. The input to each of these networks is based on an ellipticity pair at adjacent frequencies (Figure 16.1). Three inputs were used for each network, the logarithm of the absolute value of the first ellipticity (lower frequency) and the logarithm of the absolute value of the second ellipticity (higher frequency), and the sign of the difference between the first two inputs (+1 for positive and -1 for negative). These inputs are mapped to the logarithm of the half-space resistivity, which is our only output (Fig. 16.1). Logarithmic scaling avoids problems with data that span large ranges. Neural networks require all inputs to be scaled to the range [0,1] or [-1,1]. We will discuss in detail the training of the piecewise half-space neural networks for a Tx-Rx separation of 32 m. Details for other configurations are discussed in Birken and Poulton (1995).

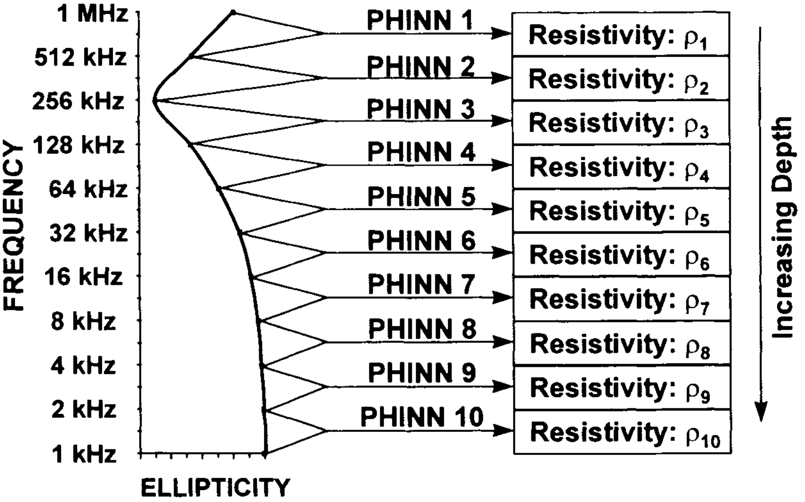

1 MHz
512 kHz
256 kHz
128 kHz
64 kHz
32 kHz
16 kHz
8 kHz
4 kHz
2 kHz
1 kHz
FREQUENCY
ELLIPTICITY
PHINN 1
PHINN 2
PHINN 3
PHINN 4
PHINN 5
PHINN 6
PHINN 7
PHINN 8
PHINN 9
PHINN 10
Resistivity: ρ_1
Resistivity: ρ_2
Resistivity: ρ_3
Resistivity: ρ_4
Resistivity: ρ_5
Resistivity: ρ_6
Resistivity: ρ_7
Resistivity: ρ_8
Resistivity: ρ_9
Resistivity: ρ_{10}
Increasing Depth

Figure 16.1. Schematic diagram of how an ellipticity sounding with 11 frequencies is decomposed into a resistivity pseudo-section by using piecewise half-space interpretation neural networks (PHINN).

The RBF neural network architecture used for the training is shown in Figure 16.2. We used a four-layer architecture where the three inputs feed to a hidden layer of RBFs, which are connected to a second back propagation hidden layer, and the output layer. The number of PEs in the hidden layers vary according to Table 16.1. For the supervised training phase, a learning-rate of 0.9, a momentum of 0.6 and the generalized delta-learning rule were used. The second hidden layer was activated by a hyperbolic tangent transfer function and the activation function of the output was a linear function.

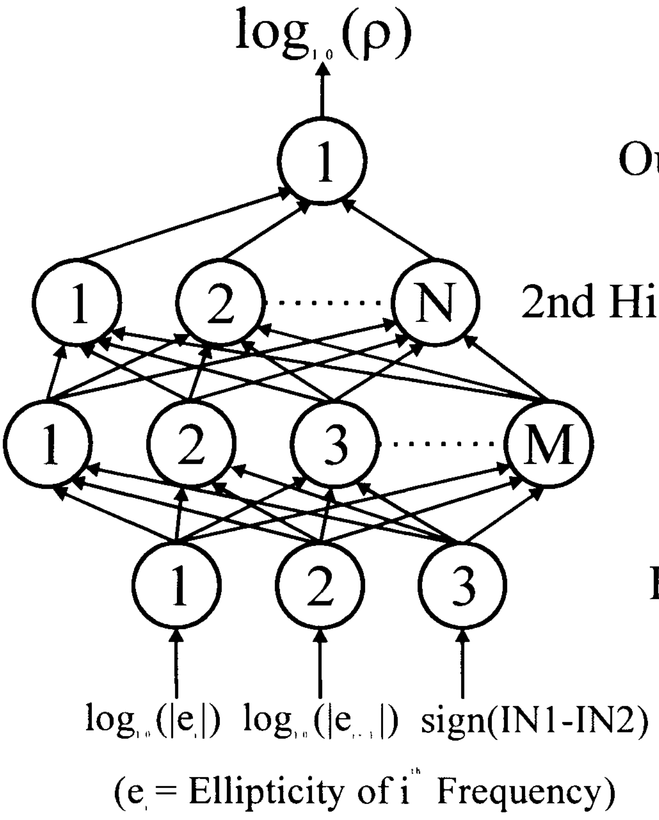

$\log_{10}(\rho)$
1
Output Layer
1
2
N
2nd Hidden Layer
1
2
3
M
RBF Layer
1
2
3
Input Layer
$\log_{10}(|e_i|)$ $\log_{10}(|e_{i+1}|)$ sign(IN1-IN2)
(e_i = Ellipticity of i^{th} Frequency)

Figure 16.2. Network architecture of RBF network used for training of the piecewise half-space neural networks for a Tx-Rx separation of 32m. IN1 and IN2 are the first two input values, N the number of RBF layer processing elements and M the number of the processing elements in the second hidden layer.

The training and test sets were created using a forward modeling code based on a program written by Lee (1986) and modified by Thomas (1996). We calculated ellipticities for 50 resistivities per decade for half-space models in our resistivity range of interest from 1 Ω·m to 10,000 Ω·m, and for 20 resistivities per decade in the test set. During the optimization of the training, I made several observations.

1) Using one decade more on each end of the resistivity range of interest improves the accuracy of the trained networks within the range of interest (Tables 16.1 and 16.2), especially for the first and the last piecewise half-space neural network. This is consistent with the known observation that most neural networks tend to have more problems in approximating

the mapping function at the ends of the interval bounding the output range. Therefore, we used model data from 0.1 Ω·m to 100,000 Ω·m as inputs, but tested the neural networks just in the range of interest.

2) Originally just the difference between the first two inputs were used as a third input, but it appeared when using field data that the neural networks are much more robust when using the sign of the difference instead. Otherwise the networks were giving too much weight to the actual value of the slope, while just the direction of the slope appears to be important.

3) The number of RBF hidden processing elements is very important for the performance of the network. Unfortunately I was not able to observe any consistent patterns in how to determine a good number, except by 'trial-and-error'.

4) The second hidden layer doesn't improve the training itself. It makes the networks more robust to noise in the field data.

The training of one piecewise half-space neural network error takes about two minutes on a 90 MHz Pentium computer depending on the network size (number of nodes) and the number of iterations to reach a sufficiently low RMS-error. To interpret one dataset with the previously trained neural network takes much less than one second.

Table 16.1
RBF neural network architecture and training parameters for the training of the piecewise half-space neural networks for a Tx-Rx separation of 32 m

Piecewise half-space neural network (PHNN)	Network input using ellipticities from following frequencies (kHz)	Number of hidden radial basis function processing elements	Number of second hidden layer processing elements	Iterations
1	973 and 1.945	35	3	45000
2	1.945 and 3.891	50	12	30000
3	3.891 and 7.782	40	3	95000
4	7.782 and 15.564	40	3	95000
5	15.564 and 31.128	40	3	45000
6	31.128 and 62.256	40	3	90000
7	62.256 and 124.512	50	12	90000
8	124.512 and 249.023	50	12	45000
9	249.023 and 498.046	40	15	55000

Table 16.2
Rms training errors for each piecewise half-space network

PHNN	Frequencies (kHz)	rms error training (0.1 to 100,000 Ohm·m)	rms error training (1 to 10,000 Ohm·m)	rms error testing (1 to 10,000 Ohm·m)
1	0.973 - 1.945	0.02623	0.01242	0.01239
2	1.945 - 3.891	0.02109	0.01561	0.01559
3	3.891 - 7.782	0.02396	0.02273	0.02300
4	7.782 - 15.564	0.02120	0.01961	0.01951
5	15.564 – 31.128	0.02163	0.01971	0.01981
6	31.128 – 62.256	0.02209	0.01997	0.01993
7	62,.256– 24.512	0.02012	0.01608	0.01595
8	124.512- 249.023	0.01908	0.01786	0.01755
9	249.023- 498.046	0.44381	0.02761	0.02748

4. CASE HISTORY

To demonstrate the capabilities of these networks they were to an interpretation with a nonlinear least-squares inversion algorithm (Dennis et al., 1981) for an example case history. A survey was conducted near Rock Springs, Wyoming, USA at the site of an abandoned underground coal mine. The goal of this investigation was to provide subsurface information about areas of subsidence, which were believed to be caused by an underground coal-seam fire. The exact location of the fire, its depth, and heading were not known. Smoke was visible on the surface in some areas where fractures had allowed the fire to vent. The fire was believed to have started in a surface outcrop as a result of a lightning strike and then spread to the seams underground.

Our investigations were performed along three east-west lines as shown in Figure 16.3. The estimated boundary of the mine fire was based on previously conducted geophysical surveys and surface observations (Hauser, 1995, personal communication). We conducted the electromagnetic survey with a 32 m Tx-Rx separation, along line 3S from 154 to 54 m, along 2S from 284 to 49 m, and along the baseline from 284 to -96 m. Stations were 5 m apart for the line 3S and 10 m for the baseline. On line 2S we started out with stations 5 m apart and switched to a 10 m station interval at station 190 m of this line. The general elevation of the site is about 2,000 m and the whole survey area slopes gradually downward to the west. The baseline drops steeply to a wash (5 m drop) between stations -20 and -50 m.

The general stratigraphy at the site shows approximately 15 m of overburden consisting of sandstones and siltstones with some shale. A thin, rider coal seam exists underneath, approximately 9 m above the main coal seam, which is 2 to 4 m thick.

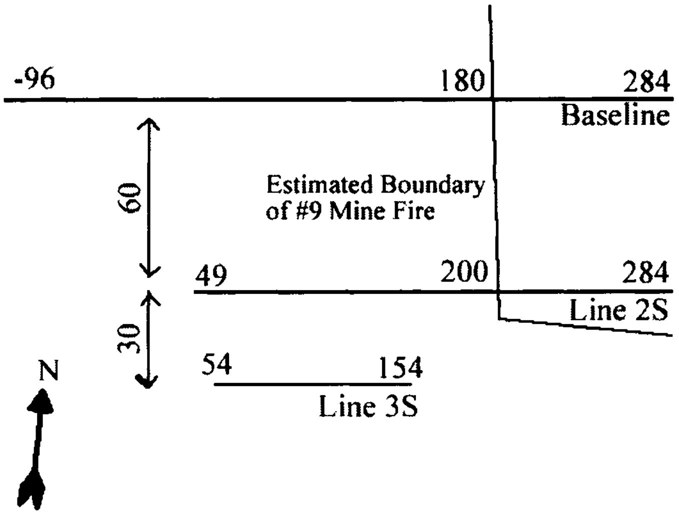


Figure 16.3. Survey lines for the #9 mine ellipticity survey area, including an estimated southwest boundary of the underground mine fire (Hauser, 1995, personal communication).

4.1 Piecewise half-space interpretation

After eliminating stations with bad data quality, we ended up with 16 out of 21 stations along line 3S, 32 of 34 for 2S, and 36 of 39 for the baseline. The highest two frequencies of 500 kHz and 1 MHz did not record usable data throughout the survey and had to be discarded. We ran the neural networks on the remaining field data, which provided us with half-space resistivity estimates for ellipticity pairs. To create comparable resistivity sections for the interpretation using a non-linear least-square inversion technique (Dennis et al., 1981), we inverted the same ellipticity pairs of adjacent frequencies for a half-space resistivity. Using the frequency and resistivity values we calculated a 'real depth' for every data point, based on depth of penetration model calculations for the ellipticity system (Thomas, 1996). We were able to plot resistivity-depth sections for each line (Figures 16.4b, 16.5b and 16.6b), based on the piecewise half-space neural network interpretation and comparable sections (Figures 16.4a, 16.5a and 16.6a), based on the inversion results. All six sections were created with the same gridding and contouring algorithm.

Line 3S was believed to be outside the subsidence and underground mine area (Figure 16.4), so we considered the resistivities shown in the resistivity sections in Figure 16.4 to be background resistivities. Therefore it was assumed that the resistivities of 40 to 55 Ω·m represent undisturbed ground (without subsidence). The inversion and neural network interpretations were nearly identical; the top portions are around 40 Ω·m, while a slightly more resistive ground of 55 Ω·m showed up between stations 54 and 100 m at a depth of 9.5 m. With this information, the west half of the resistivity sections for line 2S (Figure 16.5) also showed an area without subsidence, while the east part of line 2S was more conductive (15 to 25 Ω·m). It was believed that this was due to higher moisture content in a fractured subsidence area. Surface fractures were observed in the low resistivity areas. The boundary between the interpreted subsidence area and undisturbed ground in both sections correlated well with the previously estimated boundary (Figure 16.3) by Hauser (1995, personal communication). Comparing the resistivity sections of the baseline (Figure 16.6), it could be concluded that both interpretation techniques showed very similar results. The baseline results showed an undisturbed area in the center of the line from stations -40 to 170 m. Both sections (Figures 16.6a and 16.6b) indicated two potential subsidence areas between 170 and 270 m, and from -40 m to the east. The first subsidence area was assumed to be due to the underground mine fire and had some visible surface effects, while the second one corresponded to a topographic low with signs of ponded water in the past. A deeply incised drainage began at the west end of the line.

This comparison showed that neural networks are capable of giving an equivalent result to inversion, but in a fraction of the time. An interpretation-time comparison between the neural network and the inversion techniques, on the same 90 MHz Pentium computer, showed that the neural networks needed less than one minute to estimate the resistivities for all 84 stations, while the inversions ran for an average 5 s for one ellipticity pair or 63 min for all 84 stations. As problems move to more complex 1-, 2- or 3D cases the inversion will need much more computation time, while the trained neural network will still give an answer within seconds. Generating training sets, however, for the neural network does take significantly longer for layered-earth models.

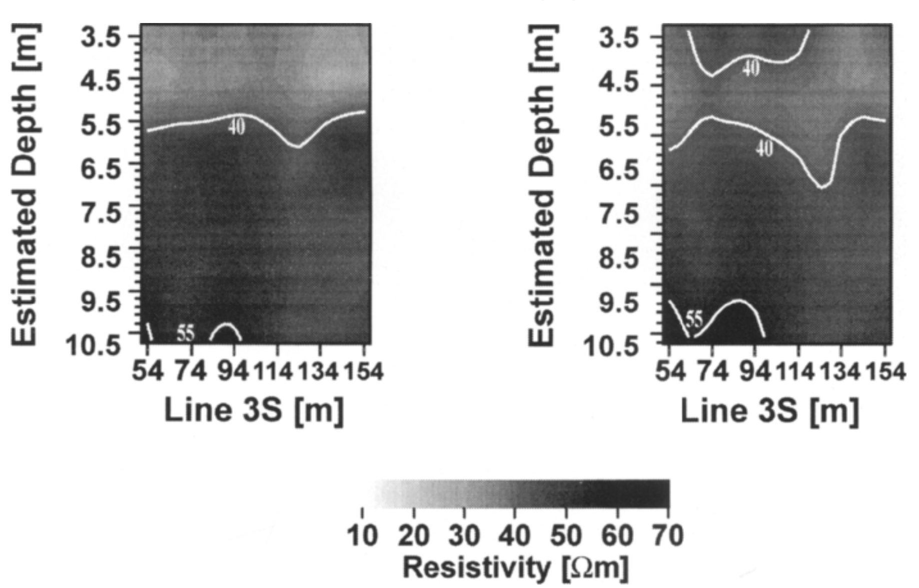


Figure 16.4. Resistivity-depth sections for line 3S (background line) created from (a) piecewise inversion results and (b) piecewise half-space resistivity interpretation neural networks results. Depth estimated by depth of investigations algorithm from Thomas (1996).

Table 16.3

Comparison of resistivity results for two selected stations for Line 3S of the Wyoming dataset

Line 3S

	Station at 124 m		Station at 64 m	
PHNN	Network Resistivity Ω·m	Inversion Resistivity Ω·m	Network Resistivity Ω·m	Inversion Resistivity Ω·m
1	43.2	42.8	48.2	47.6
2	41.9	42.7	47.3	47.9
3	39.8	41.7	45.3	48.0
4	37.7	41.0	44.7	47.2
5	36.7	39.7	47.2	46.7
6	32.9	35.7	38.8	45.7
7	28.4	24.9	28.7	25.1
8	41.8	35.9	43.4	36.9
Half-space	41.1	41.7	45.1	47.1

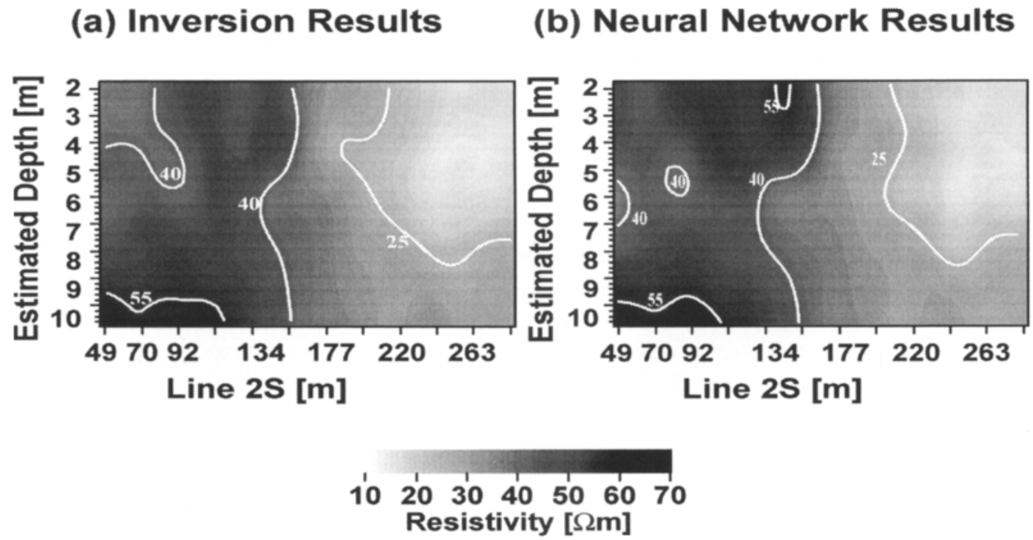


Figure 16.5. Resistivity-depth sections for line 2S created from (a) piecewise inversion results and (b) piecewise half-space resistivity interpretation neural networks results. Depth estimated by depth of investigations algorithm from Thomas (1996).

Table 16.4
Comparison of resistivity results for two selected stations for Line 2S of the Wyoming dataset

Line 2S

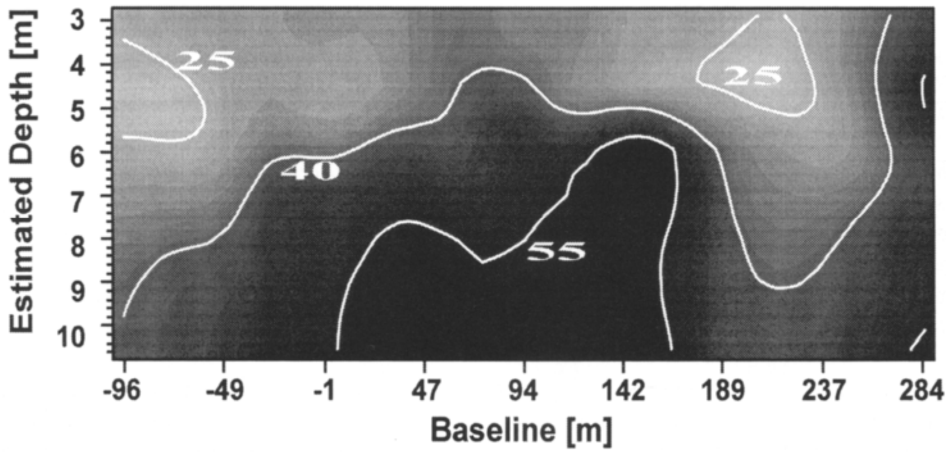
PHNN	Station at 244 m Network Resistivity Ω·m	 Inversion Resistivity Ω·m	Station at 69 m Network Resistivity Ω·m	 Inversion Resistivity Ω·m
1	25.3	25.3	55.5	55.6
2	23.5	51.1	51.3	41.9
3	19.2	19.8	46.4	49.1
4	10.6	12.9	45.3	47.7
5	11.5	10.7	47.5	46.9
6	18.2	14.9	37.2	44.0
7	15.6	15.5	31.0	27.2
8	16.7	16.0	58.4	51.7
Half-space	23.6	20.4	51.5	52.7

Table 16.5
Comparison of resistivity results for two selected stations for the baseline of the Wyoming data set

Baseline

PHNN	Station at 174 m Network Resistivity Ω·m	 Inversion Resistivity Ω·m	Station at 54 m Network Resistivity Ω·m	 Inversion Resistivity Ω·m
1	39.5	40.0	60.6	61.0
2	51.9	39.9	61.2	61.1
3	44.6	47.3	59.1	59.6
4	43.5	46.2	60.0	59.0
5	43.7	44.4	58.4	55.6
6	36.8	42.8	35.2	42.8
7	20.9	19.8	36.8	32.1
8	40.7	33.9	47.3	39.9
Half-space	42.5	42.3	55.7	59.3

(a) Inversion Results



(b) Neural Network Results

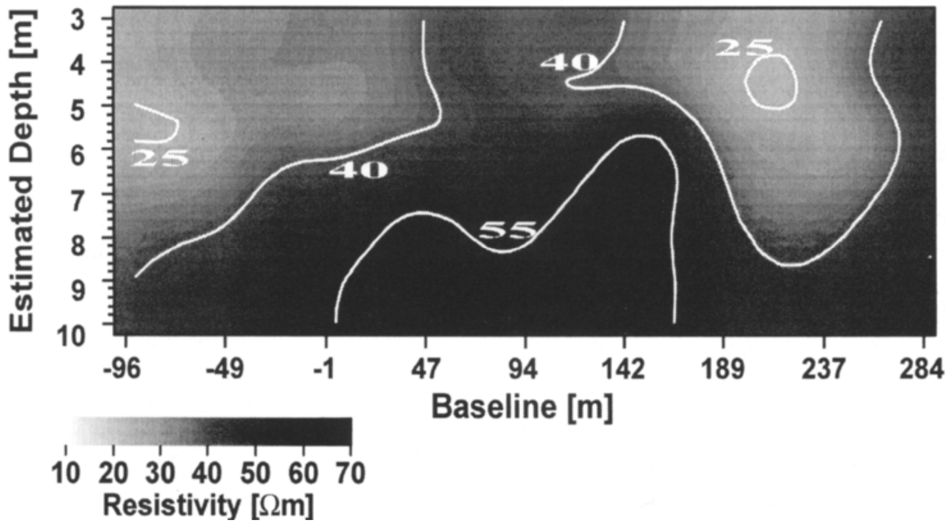


Figure 16.6. Resistivity-depth sections for the baseline created from (a) piecewise inversion results and (b) piecewise half-space resistivity interpretation neural networks results. Depth estimated by depth of investigations algorithm from Thomas (1996).

4.2. Half-space interpretations

One half-space neural network was trained for each Tx-Rx separation. The inputs were the 10 ellipticity values at the recording frequencies. They were scaled by taking the logarithm of the absolute value of the ellipticity. These inputs were mapped to the logarithm of the half-space resistivity, which is the only output. Training of the 32 m Tx-Rx separation half-space neural network is discussed in this section.

A RBF neural network architecture contained 10 inputs, 35 hidden RBF processing elements, 3 second hidden layer processing elements, and the resistivity output processing elements. For the training a learning-rate of 0.9, a momentum of 0.6 and the delta-learning rule were applied. The second hidden layer used a hyperbolic tangent activation function and the output processing elements used a linear transfer function. The training and test sets were the same as for the piecewise half-space neural network discussed above. After 40,000 iterations the overall rms error was down to an acceptable 0.01706. The rms errors for the range of interest were 0.00386 for the training set and 0.00411 for the testing set. The same 4 observations made during the training of the 32 m piecewise half-space neural networks (see above) were found to apply to the 32 m half-space network training.

Both the half-space and the piecewise half-space neural networks were trained on the same dataset, but the capabilities of the networks are quite different. One disadvantage of the half-space neural network is that an incomplete ellipticity sounding curve, e.g. due to a system problem at just one frequency, leads to a significant error in the half-space resistivity. The piecewise neural networks are more flexible, since they require only two adjacent ellipticity readings. A comparison between both half-space neural network interpretations is shown in Figure 16.7 for the 124 m station of line 3S. The piecewise half-space neural networks (RMS = 0.000045) fit the field data better than the half-space neural network (RMS = 0.000132). For this example, it fits the field data better than the piecewise inversion result (RMS = 0.000069). In every instance, when each sounding was inverted as a layered-earth model, the inversion converged to a half-space model. A great deal of consistency was observed between the piecewise neural network and inversion results as shown in Tables 16.3 to 16.5. Two example stations of each line are shown and the estimated resistivities of both techniques are

5. CONCLUSION

The half-space modules of the neural network interpretation system were successfully tested on field data from a survey over a subsiding underground coal mine. The neural network produced resistivity estimates that were in very close agreement with results from a non-linear inversion routine. RBF networks were able to produce more accurate results than back-propagation, especially when trained on a resistivity range that extended one decade beyond the resistivity range expected to be encountered in most situations.

A RBF neural network trained to interpret ellipticity data from 1 kHz to 1MHz at a 32 m Tx-Rx separation cannot interpret magnetic field components from a different frequency range or coil separation. For half-space resistivities, re-training a network for different parameters can be accomplished in a few minutes. The actual interpretation times for the whole Wyoming dataset showed a 60-times faster computing time in favor of the neural networks. The speed advantage offered by the neural networks makes them applicable where near real-time or in-field estimations are required. Neural networks should be considered a complimentary tool for other interpretation techniques. The neural networks find a model that best fits the test data based on models used for training. The result from a neural network can be used as a starting model for inversion to decrease inversion times.

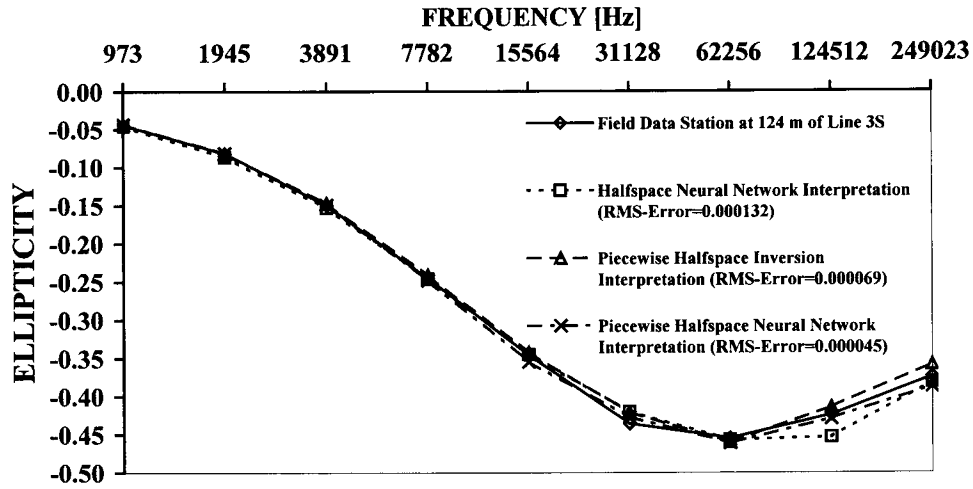


Figure 16.7. Comparison of data fits for the 124 m station of line 3S for inversion, half-space neural network, piecewise half-space neural network, and field data. In addition I show a comparison of the estimated half-space resistivities using ellipticities at the lower 9 frequencies for the inversion calculation and for the half-space neural network. The starting model for all inversions was 40 Ω·m.

REFERENCES

Bak, N., Sternberg, B., Dvorak, S., and Thomas, S., 1993, Rapid, high-accuracy electromagnetic soundings using a novel four-axis coil to measure magnetic field ellipticity: J. Appl. Geophys., **30**, 235-245.

Birken, R., 1997, Neural network interpretation of electromagnetic ellipticity data in a frequency range from 1 kHz to 32 MHz: Ph.D. Thesis, University of Arizona.

Birken, R., and Poulton, M., 1995, Neural network interpretation scheme for high and medium frequency electromagnetic ellipticity surveys: Proceedings of the SAGEEP '95, 349-357.

Bishop, C., 1995, Neural Networks for Pattern Recognition: Oxford Press.

Broomhead, D., and Lowe, D., 1988, Multivariable functional interpolation and adaptive networks: Complex Systems, **2**, 321-355.

Darken, C., and Moody, J., 1990, Fast adaptive K-means clustering: Some empirical results: IEEE INNS International Joint Conference on Neural Networks, 233-238.

Dennis, J., Gay, D., and Welsch, R., 1981, An adaptive nonlinear least-squares algorithm: ACM Transactions on Mathematical Software, **7**, 3, 348-368.

Geman, S., Bienenstock, E. and Doursat, R., 1992, Neural networks and the bias/variance dilema: Neural Computation, **4**, 1-58.

Girosi, F. and Poggio, T., 1990, Networks and the best approximation property: Biological Cybernetics, **63**, 169-176.

Haykin, S., 1994, Neural Networks. A Comprehensive Foundation: Macmillan.

Hertz, J., Krogh, A. and Palmer, R.G., 1991, Introduction to the Theory of Neural Computation: Addison Wesley.

Lee, K., 1986, Electromagnetic dipole forward modeling program, Lawrence Berkeley Laboratory, Berkeley, CA.

Light, W., 1992, Some aspects of radial basis function approximation, *in* Singh, S., Ed., Approximation Theory, Spline Functions and Applications: NATO ASI series, **256**, Kluwer Academic Publishers, 163-190.

Michelli, C., 1986. Interpolation of scattered data: distance matrices and conditionally positive definite functions: Constructive Approximations, **2**, 11-22.

Moody, J., and Darken, C., 1989, Fast learning in networks of locally-tuned processing units: Neural Computation, **1**, 281-294.

Musavi, M., Ahmed, W., Chan, K., Faris, K., and Hummels, D., 1992, On the training of radial basis function classifiers: Neural Networks, **5**, 595-603.

Poggio, T. and Girosi, F., 1989, A theory of networks for approximation and learning: A.I. Memo No. 1140 (C.B.I.P. Paper No. 31), Massachusetts Institute of Technology, Artificial Intelligence Laboratory.

Poggio, T. and Girosi, F., 1990a, Regularization algorithms for learning that are equivalent to multilayer networks. Science, **247**, 978-982.

Poggio, T. and Girosi, F., 1990b, Networks for approximation and learning. Proceedings of the IEEE, **78**, 1481-1497.

Powell, M., 1987, Radial basis functions for multivariable interpolation: A review, *in* Mason, J. and Cox, M., Eds., Algorithms for Approximation: Clarendon Press.

Spath, H., 1980, Cluster Analysis Algorithms for Data Reduction and Classification of Objects: Elis Horwood Publishers.

Sternberg, B., and Poulton, M., 1994, High-resolution subsurface imaging and neural network recognition: Proceedings of the SAGEEP '94, 847-855.

Thomas, S., 1996, Modeling and testing the LASI electromagnetic subsurface imaging systems: Ph.D. Thesis, University of Arizona.

Tikhonov, A. and Arsenin, V., 1977, Solutions of Ill-Posed Problems: W.H.Winston.

Zell, A., 1994, Simulation Neuronaler Netze: Addison Wesley.

Chapter 17
Extracting IP Parameters from TEM Data

Hesham El-Kaliouby

1. INTRODUCTION

The identification of materials by their electrical properties is effective since the electrical properties vary for different earth materials by over 28 orders of magnitude (Olhoeft, 1985). Significant differences between the properties of different materials exist throughout the electromagnetic spectrum and especially at the lower frequencies used for geophysical investigations. Hence, electrical and electromagnetic methods can be used as diagnostic tools for geophysical prospecting by identifying the electrical properties of the target (e.g. fluids, minerals or trash).

The electrical methods (e.g. DC resistivity and IP) rely on applying a voltage into the ground through a series of metallic electrodes (stakes) pounded into the ground and then measuring the current produced. To measure IP effects, you cycle the transmitter on and off and measure the voltage decay in the ground while the transmitter is off. IP methods are designed intentionally to detect the dispersion (change with frequency) of electrical properties of the earth materials that occur at audio frequencies and lower. Induced electrical polarization (IP) is a good indicator of heterogeneous materials. Rocks and minerals (e.g. iron ores, sulfides, clays, and graphite, etc.) are typical examples of these materials. The electrical properties of such materials exhibit complex resistivity in the low frequency range. The complex resistivity can be represented by different models such as the Cole-Cole model, which is a curve that fits electrical parameters (chargeability (m), time constant (τ), frequency parameter (c) and DC conductivity (σ)) to the measured (or calculated) voltage response from the earth (or geologic model) (see Figure17.1).

In the electromagnetic methods, systems are designed based on the concept that the excited EM fields in the ground generate eddy currents, which can be detected in terms of secondary magnetic fields accompanying these currents. Such EM methods do not generally rely on using electrodes (they use loops of wire placed on the surface); thus, they bypass the problems related to the use of electrodes such as poor coupling with the ground, poor signal strength, noise, high labor cost to install them and problems arising when the upper layer in the ground behaves as an electrical insulator (such as very dry soils). Since the IP technique provides very useful information about the properties of geologic materials and the TEM technique provides better field data acquisition, it is important to study the extraction of the IP information from the TEM data. The knowledge of the electrical behavior of the heterogeneous materials helps greatly in improving the accuracy of the interpretation.

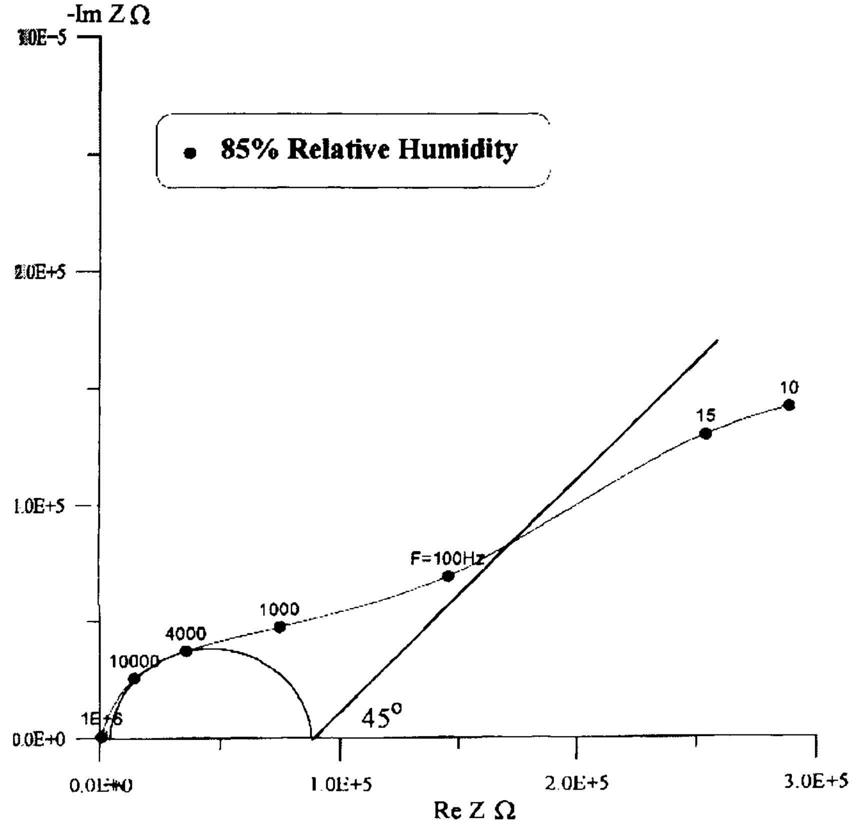


Figure 17.1. Results of samples, measured at a moisture content equivalent to 85% relative humidity, represented in the impedance plane. The semi-circle shows the Cole-Cole fit to the data. The 45-degree line shows the power law fit to the data. Note that by using only the Cole-Cole or power law model, the fit to the data is very poor.

The time-domain EM (TEM) response measured using a coincident loop system above a dispersive conductive earth can show evidence of IP effects that manifest themselves as a negative response (NR) phenomenon where the transient voltage response undergoes a rapid decay, which is followed by a polarity reversal (Figure 17.2). The negative response is regarded as noise by many practicing geophysicists and eliminated from their field data because there exists no convenient way of inverting the IP effect. Hence geophysicists are forced to throw away valuable data that contains information on the electrical properties of the earth material being surveyed.

The negative response in TEM data occurs because the inductive (positive) current excited by the loop in the ground charges the polarizable ground and when the inductive current decays, the ground discharges with its longer time constant (polarization current) leading to the negative response (Flis et al., 1989). This phenomenon may be used to detect the underground polarizable targets (Lee, 1981; Smith and West, 1988; El-Kaliouby et al., 1995, 1997). The electrical properties of the polarizable target (e.g. groundwater and conducting minerals) and loop radius of the excitation current source play an important role in determining the onset of the negative response and its magnitude. For example, the electrical properties of a clay-water mixture, have a strong role in determining the onset of the negative response and its magnitude, and hence can be used as an indictor of the presence of groundwater. The main properties that affect the detection of the negative response of clay-bearing rock are the moisture content, grain size, solution conductivity and the clay content.

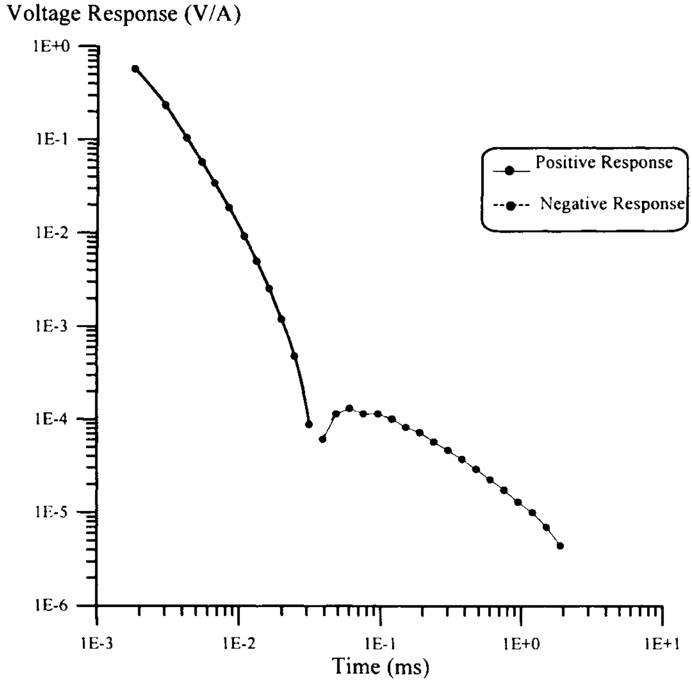


Figure 17.2. Measured transient voltage showing the negative response phenomenon.

Much research has been done on the inversion of EM measurements above polarizable ground. A number of methods have been used for determining the chargeability from the time-domain or frequency-domain IP data using an electrode system (Sumner, 1976). Inversion methods have been developed for estimating the Cole-Cole parameters from time-domain IP data (Johnson, 1984; Oldenberg, 1997).

In this work, a neural network approach is presented for finding the electrical properties of half-space and layered polarizable conducting targets using transient electromagnetic coincident or central loop systems to predict the Cole-Cole parameters mentioned above.

2. FORWARD MODELING

The half-space models are coded based on an equation derived from the late time voltage response by Lee (1981). This equation is based on obtaining a low-frequency series expansion for the frequency response of the induced voltage.

The transient voltage is obtained for the layered earth models by applying the inverse Fourier transform for the frequency response function. This function is obtained at each frequency by evaluating the inverse Hankel transform. Both the inverse Fourier transform and inverse Hankel transform integrals are evaluated using a linear digital filter algorithm based on the work of Anderson (1982).

At the heart of the forward modeling is the complex resistivity model for a polarizable earth, which can be described by a Cole-Cole model. The Cole-Cole model (or other similar models) is a mathematical model, which is used to describe the complex resistivity in terms of the electrical parameters namely, chargeability, time constant, frequency parameter, and DC conductivity. The Cole-Cole model is described by the following equation (Pelton, et al., 1978):

$$\sigma(\omega) = \sigma_o \left[1 + (i\omega\tau)^c\right] / \left[1 + \alpha(i\omega\tau)^c\right], \tag{17.1}$$

where $\sigma(\omega)$ is the complex conductivity at a frequency ω, σ_0 is the DC conductivity, τ is the time constant, c is the frequency parameter and $\alpha=1-m$ where m is the chargeability and is given by $m = 1 - \left(\frac{\sigma_o}{\sigma_\infty}\right)$. (17.2)

The Cole-Cole model is a simple relaxation model that has been found to fit a variety of laboratory complex resistivity results (Pelton, et al., 1978). Cole and Cole (1941) originally proposed the model to predict complex dielectric behavior. The parameters of this model may be related to physical rock properties and it can be used to generate many other popular impedance models such as the Debye model.

3. INVERSE MODELING WITH NEURAL NETWORKS

Computational neural networks have been used before to invert the electrical parameters of a layered earth (resistivity and thickness for each layer) using frequency domain data (see Chapter 14; Poulton and Birken, 1998). In this study, the neural network was designed to learn to extract the Cole-Cole parameters from the input voltage-time data of half-space and two-layer polarizable ground. The network was trained using the modular neural network (MNN) architecture (see Chapter 15 for description of MNN). The input layer has as many input nodes as there are input voltage samples with time. The decay curve was sampled from 1 μs to 1 second and used five voltage samples per decade as the input pattern. There are four output nodes in the output layer for the half-space case (m, τ, c and σ_0) and three output nodes for the case of two-layer earth model (m, τ and c). The MNN had five local experts with 7 hidden PEs each. The tanh function was used as the activation function. The network was trained for 50,000 iterations.

Regardless of the method used for the inversion of geophysical data, equivalence or ambiguity remains a problem. Equivalence results when different earth models result in nearly the same response due to the non-uniqueness of the measurement. The equivalent models lead to ambiguity in the interpretation because of a lack of sensitivity of the measurement to changes in the model parameters. In this study we found that ambiguity decreases when the magnitude of the negative voltage becomes large. This may be realized by using a loop radius that leads to the largest negative response (El-Kaliouby et al., 1997) in the mid-range of the Cole-Cole parameters for which training is made. High values of the chargeability help in resolving the ambiguity since they lead to a stronger negative response. The choice of the time range within which the voltage is sampled also improves the results. When the time range contains nearly equal contribution from the positive part and the negative part of the voltage response, better results are obtained.

The data from loops of two different radii (dual loops) produced lower errors since there were fewer problems with equivalence. Decomposition of the parameter ranges into two segments enhanced the accuracy since it reduced the ambiguity probability.

As discussed in Chapter 4, a neural network will typically give an average response when presented with equivalent training models. The goal of this chapter is to determine the quality of the network-estimated model and, when an error is observed, be able to attribute the error either to network training or to equivalence problems.

4. TESTING RESULTS

4.1. Half-space

The network was trained for the cases in which the voltage response contained a single sign reversal. The network was trained for different ranges of m, τ, c and σ_0. Based on the training set error, the ranges of the inversion parameters with the lowest rms error (below 15%) were: m=[0.1 - 0.5]; τ=[10^{-1}ms - 10^{2}ms]; c=[0.2 - 0.8] and σ_o =[10^{-4}S/m-10^{-1}S/m] within a sampling time period of [10^{-4}ms-10^{4}ms] which cover the time windows of all the current TEM field systems. The network was trained for different loop radii [10 m-100 m], which also fall within the practical ranges of measurements in fieldwork. It was found that the rms error ranged between 6%-15% with the different loop radii.

To improve the inversion, the voltage response of loops of two different radii were used together for the same set of parameters (m, τ, c and σ_0) to resolve the ambiguity that may arise in the single loop data. In this case, the numbers of input and hidden layer nodes were doubled. When using the voltage response of 100-m and 50-m loops, the rms error was only 9% while using 50-m and 25-m loops, resulted in an rms error as low as 5% which is a very good improvement in the inversion results (Table 17.1).

Table 17.1
Half-Space rms error for single and two (dual) loops for the parameter ranges: m=[0.1-0.5]; τ=[10^{-1}ms- 10^{2}ms]; c=[0.2-0.8]; σ_0 =[10^{-4}S/m-10^{-1}S/m] and sampling time period of [10^{-4}ms-10^{4}ms]

Model	Loop Type	rms error (%)
Half-Space	Single Loop (10m-100m)	6-15
	Dual Loop	5-9

4.2. Layered ground

After studying each of the layered-earth parameters, it was found that due to current channeling, the magnitude of the negative response (NR) in layered ground could be much greater than the corresponding response of the half-space model (by 10^3 times or more) when the polarizable layer is relatively more conductive than the surrounding layers (Spies and Parker, 1984; Smith and West, 1988). In this case, the fundamental positive current decays much faster ($\alpha\ t^{-4}$) than in the half-space case ($\alpha\ t^{-5/2}$). Current channeling is mainly controlled by the conductivity contrast and the thickness of the polarizable layer.

A network was trained for a two-layer earth model. First, I inverted for the Cole-Cole parameters and the layering parameters, namely: first layer conductivity (σ_1), second layer conductivity (σ_2) and first layer thickness (h_1). Training errors were higher than desired due to the ambiguity that increases with the increase of the number of inversion parameters and the large set of training data. Next, I decided to invert only for the Cole-Cole parameters m, τ and c at different conductivities, thickness and loop radii for the two cases where the first layer is polarizable and when the second layer is polarizable. In this case, the model parameters for conductivity and thickness are assumed to be derived from some other source and used as input to the network in addition to the voltage information.

4.3. Polarizable first layer

Figure (17.3) shows the rms error of some combinations of the first and second layer conductivities for a thickness of 5 m and loop radius of 28 m, which corresponds to a loop side of 50 m. The Cole-Cole parameter ranges are: m=[0.1 - 0.5]; τ=[10^{-1}ms - 10^{3}ms]; c=[0.2 - 0.8]; σ_1 =[10^{-4}S/m-10^{-1}S/m] and σ_2=[10^{-4}S/m-1 S/m] within a sampling time period of [10^{-3}ms-10^{3}ms]. The error is only dependent on the conductivity contrast when the second (non-polarizable) layer is more conducting than the upper resistive polarizable layer. In this situation, the current escapes to the more conductive layer thus, the positive voltage decays slowly leading to a weaker negative response. The weaker response is harder to learn and leads to a higher rms error for the polarization parameters. The error is exacerbated by the already weak IP response of the thin polarizable layer. The data shown in figures 17.3 – 17.20 show the average rms errors for all the Cole-Cole parameters as a function of the conductivities of each layer. So, in Figure 17.3, if the first layer log10 conductivity is –3 and the second layer log10 conductivity is –2, the average rms error of m, τ, and c is approximately 20%. Typically, the estimated value of m has a lower rms error than τ and c.

Figures (17.4 and 17.5) show the effect of increasing the first layer thickness to 30 m and 100 m for the same aforementioned parameters and a loop radius of 28 m. The rms errors decrease because of the increasing thickness of the polarizable layer.

Figures (17.6, 17.7 and 17.8) show the effect of another loop radius of 56 m, which corresponds to a loop side of 100 m at the different combinations of the Cole-Cole parameters for the first layer thickness of 5 m, 30 m and 100 m respectively. The change of the radius has no significant effect on the training errors.

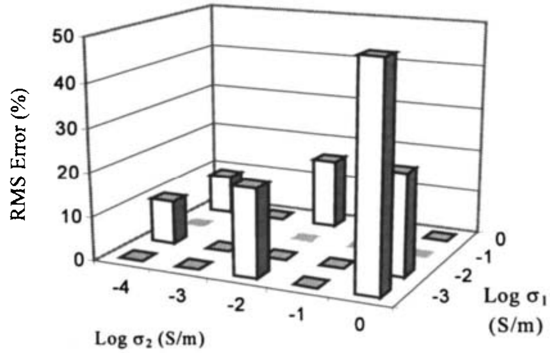


Figure 17.3. RMS error of some combinations of the first and second layer conductivities at thickness of 5 m and loop radius of 28 m.

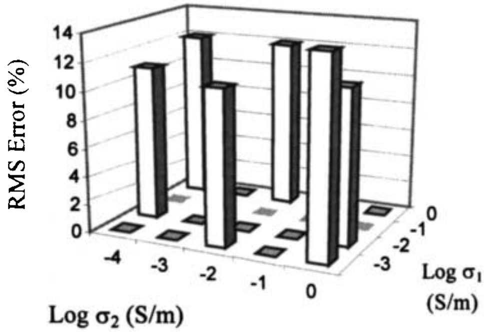


Figure 17.4. RMS error of some combinations of the first and second layer conductivities at thickness of 30 m and loop radius of 28 m.

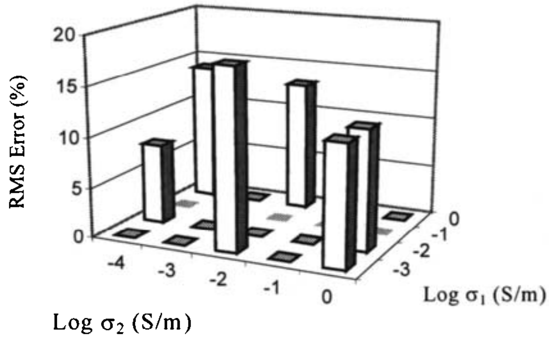


Figure 17.5. RMS error of some combinations of the first and second layer conductivities at thickness of 100 m and loop radius of 28 m.

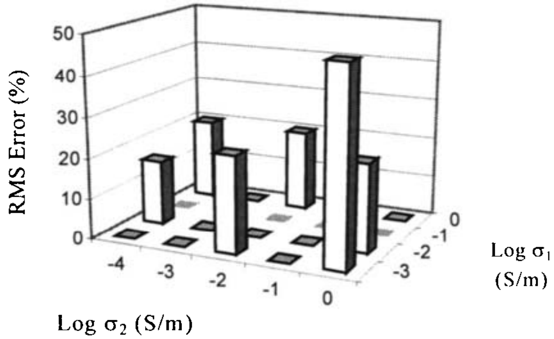


Figure 17.6. RMS error of some combinations of the first and second layer conductivities at thickness of 5 m and loop radius of 56 m.

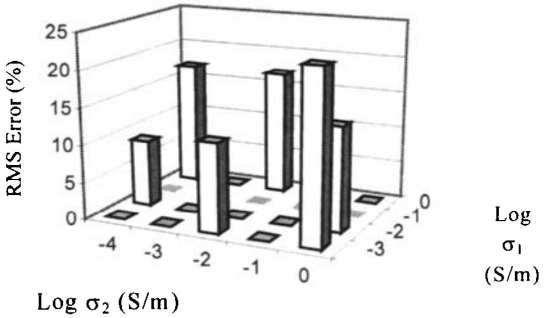


Figure 17.7. RMS error of some combinations of the first and second layer conductivities at thickness of 30 m and loop radius of 56 m.

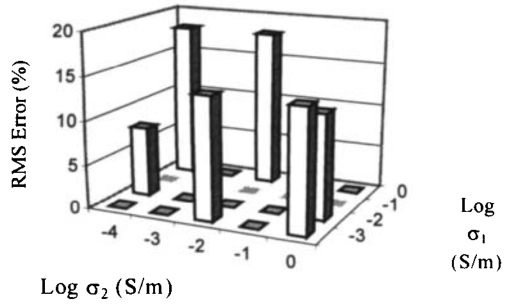


Figure 17.8. RMS error of some combinations of the first and second layer conductivities at thickness of 100 m and loop radius of 56 m.

In order to improve the inversion by resolving the ambiguity that may arise in the single loop data, the voltage responses from two loops of different radii were used for the same set of parameters. Figure (17.9) shows the dual loop radii rms error for a 5 m thickness where we used data from both of the loop radii of 28 m and 56 m. The medium-range rms errors were reduced to less than 10%. However, the dual loop radii inversion did not lead to useful improvement when the error was high.

Figures (17.10 and 17.11) show the dual loop radii results for the 30 m and 100 m thickness. The errors improve for most of the cases.

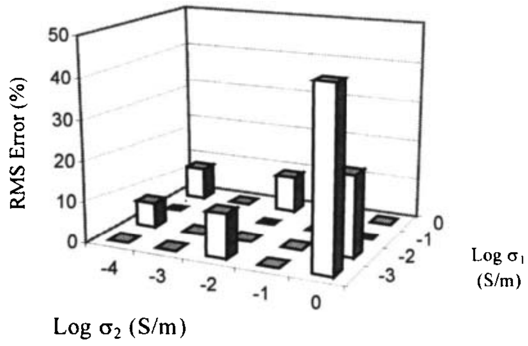


Figure 17.9. RMS error of some combinations of the first and second layer conductivities at thickness of 5 m and loop radii of 28 m and 56 m.

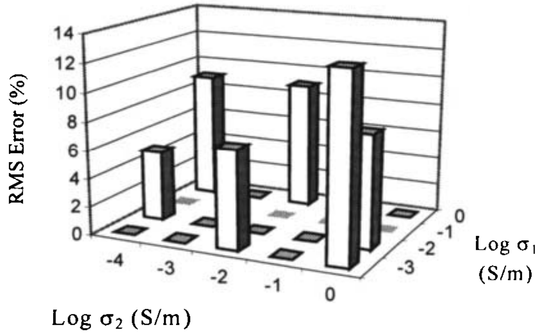


Figure 17.10. RMS error of some combinations of the first and second layer conductivities at thickness of 30 m and loop radii of 28 m and 56 m.

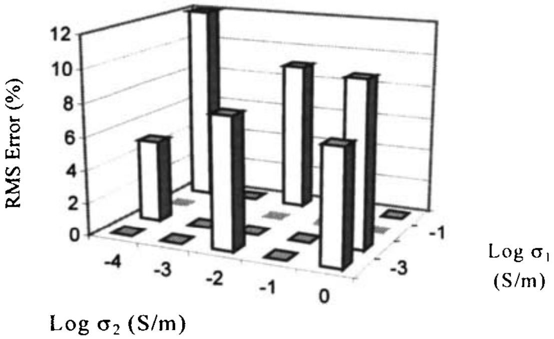


Figure 17.11. RMS error of some combinations of the first and second layer conductivities at thickness of 100 m and loop radii of 28 m and 56 m.

4.4. Polarizable second layer

Figure (17.12) shows the rms error of some combinations of the first and second layer conductivities for a layer thickness of 5 m and loop radius of 28 m at different combinations of the Cole-Cole parameters of the second layer.

Notice from the plot that the training errors are generally better than the corresponding errors in the polarizable first layer case. The small thickness of the non-polarizable layer aids in making the positive voltage decay early and thus does not degrade the negative voltage except when the polarizable layer is highly conducting (σ_2=1 S/m). We notice a slightly higher error due to the strong positive voltage related to the second conducting layer.

Figures (17.13 and 17.14) show the 30 m and 50 m thickness of the first layer. Notice that when the first non-polarizable layer is more conducting than the second polarizable one, the error is relatively high, which can be attributed to the current channeling which takes place in the first layer. The negative voltage will be weaker which will lead to poor learning or high rms error for the polarization parameters. However, for all the other cases, the training error is approximately 10%.

Figures (17.15, 17.16 and 17.17) show the effect of another loop radius of 56 m, which corresponds to a loop side of 100 m at the different combinations of the Cole-Cole parameters for the first layer thickness of 5 m, 30 m and 50 m respectively. The change of the radius has no significant effect on the training errors.

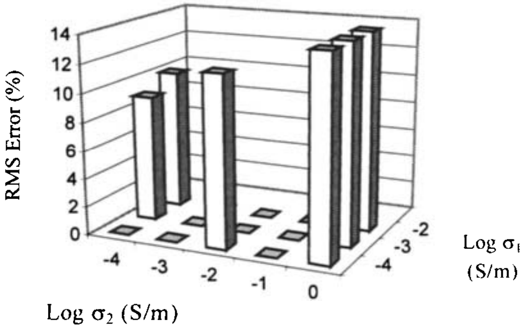


Figure 17.12. RMS error of some combinations of the first and second layer conductivities at thickness of 5 m and loop radius of 28 m.

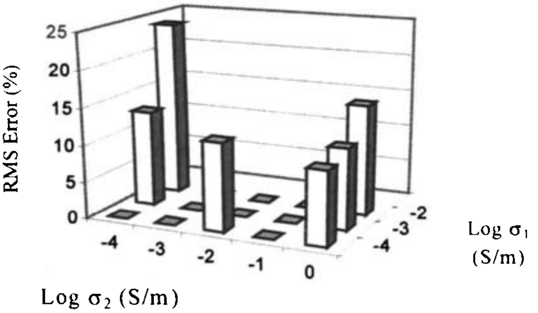


Figure 17.13. RMS error of some combinations of the first and second layer conductivities at thickness of 30 m and loop radius of 28 m.

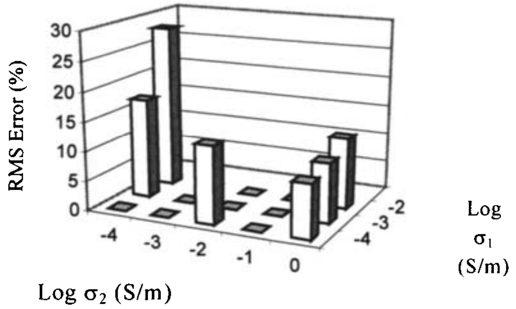


Figure 17.14. RMS error of some combinations of the first and second layer conductivities at thickness of 50 m and loop radius of 28 m.

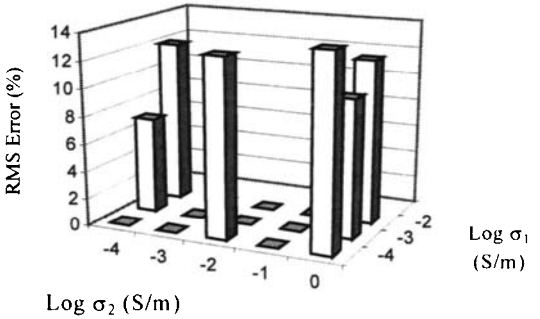


Figure 17.15. RMS error of some combinations of the first and second layer conductivities at thickness of 5 m and loop radius of 56 m.

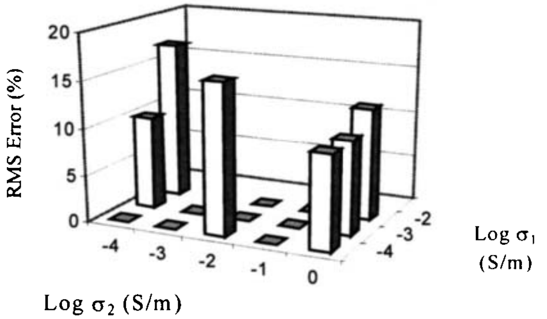


Figure 17.16. RMS error of some combinations of the first and second layer conductivities at thickness of 30 m and loop radius of 56 m.

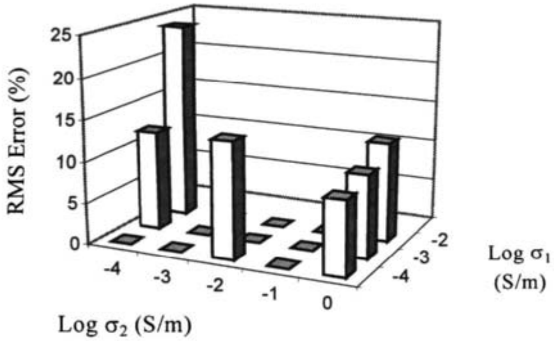


Figure 17.17. RMS error of some combinations of the first and second layer conductivities at thickness of 50 m and loop radius of 56 m.

Figures (17.18, 17.19 and 17.20) show the dual loop radii (28 m and 56 m)rms error for 5-m, 30-m and 50-m thickness. The error is reduced to less than 10% in most cases.

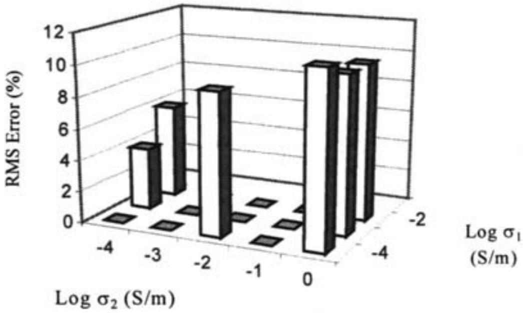


Figure 17.18. RMS error of some combinations of the first and second layer conductivities at thickness of 5 m and loop radii of 28 m and 56 m.

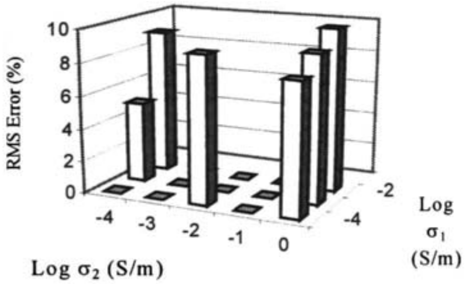


Figure 17.19. RMS error of some combinations of the first and second layer conductivities at thickness of 30 m and loop radii of 28 m and 56 m.

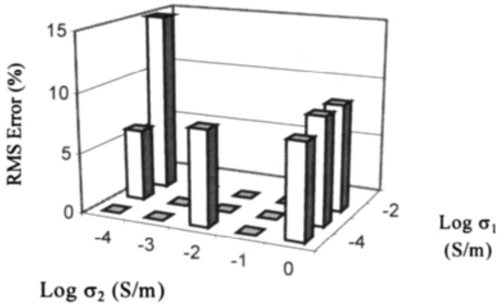


Figure 17.20. RMS error of some combinations of the first and second layer conductivities at thickness of 50 m and loop radii of 28 m and 56m.

5. UNCERTAINTY EVALUATION

To address the question of confidence or certainty in the network estimates of the Cole-Cole parameters a second network was designed that associates an error range with each estimate. It was found that errors in the network estimates were associated with the voltage response cases with ambiguity that resulted in poor learning. The error had a direct relation to the voltage response and this relation was used to predict the error ranges from the voltage response by training the network for their relation.

The input of this network was the voltage data with time while the outputs were the errors in each parameter. Those errors were expressed as ranges: <5%; 5%-10%; 10%-15%; 15%-20%; and >20%. Each interval was defined by a certain number (e.g. n=1, 2, 3, 4, and 5). The MNN network parameters were identical to the first network with the received voltage values as input but the output was an error range from 1 to 5 for each of the three Cole-Cole parameters. The error range codes were based on the training errors from the first network. For a given voltage pattern, the first network estimated values for m, τ, and c. Table 17.2 shows the cumulative frequency of accurately estimating the error range.

Table 17.2
Cumulative frequency of not missing error range by more than *n* ranges

Missed ranges (*n*)	m	τ	c
0	98.18	62.76	59.90
1 (+/- 5%)	1.56	23.18	32.81
2 (+/- 10%)	0.26	12.50	5.73
3 (+/- 15%)	0.0	1.56	1.56

Table 17.2 is used to interpret the accuracy of the error range estimate from the second neural network. For the chargeability, m, the network estimated the error range correctly

98.18% of the time. However, for 1.56% of the models the network misclassified the correct error range by one range. If the network estimated the error range as class-2 (5-10% error), it could really be a class-1 error (<5% error) or a class-3 error (10-15% error) for 1.56% of the models. Missing an error range by 1 range is equivalent to estimating the error with error bars of +/- 5%. Similarly, the estimate of the error for τ is off by two error ranges 12.5% of the time. The error range for τ would have error bars of +/- 10% for 12.5% of the models.

6. SENSITIVITY EVALUATION

The mean-squared error is used as a measure of sensitivity for the analysis. This error is referred to as a fitness measure in the following discussion. The fitness becomes zero at the correct values of the parameters. The purpose of the fitness plots is to view the sensitivity of the error, with respect to the parameters, between the measured field response and the responses corresponding to certain parameter ranges. A number of models are calculated for a variety of Cole-Cole parameters and layering parameters. The response from each combination of parameters is compared to the measured field data and the mean-squared error between the model response and field data is calculated. The sensitivity plots have a sharp negative minima when the model response matches the field data since the mean-squared error is close to zero.

If the fitness has a steep slope (high sensitivity) for a certain parameter, then it is expected that the learning error will be small and this parameter can be accurately inverted, provided that those correct parameter values are within the range of the training set, as is the case for the chargeability (*m*) and time constant (τ) in Figure 17.21.

Figure 17.21. Example of high sensitivity plot for the fitness $(V/A)^2$ vs. the chargeability (m) and the time constant (τ in sec.). Both parameters show high sensitivity. Data in this plot correspond to the field data presented in Section 7.

Low fitness means that ambiguities are expected around the correct parameters and it is expected that the learning error will be high and these parameters will be poorly inverted as in Figure 17.22.

Using this information, we can distinguish between the errors that are to the result of the learning process and the errors that are the result of the nature of ambiguity in the voltage data itself.

7. CASE STUDY

The field data used here were part of a TEM survey for ground water carried out inside the caldera of the Fogo volcano (Cape Verde islands located 600 Km from the coast of West Africa). The measurements were done using a PROTEM EM47 system with an in-loop configuration of 100 m, a turn-off time of 5.2 μsec and a current of 2 A. The coincident loop and in-loop configurations showed similar behavior at late times.

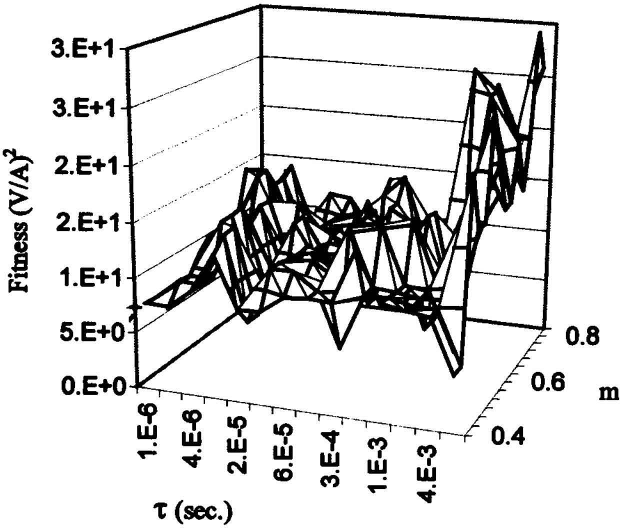


Figure 17.21. Example of high sensitivity plot for the fitness (V/A)^2 vs. the chargeability (m) and the time constant (τ in sec.). Both parameters show high sensitivity. Data in this plot correspond to the field data presented in Section 7.

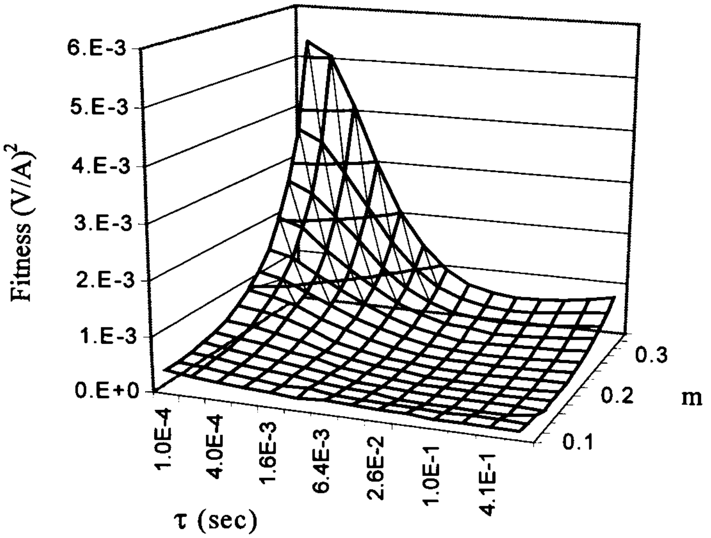


Figure 17.22. Example of low sensitivity plot for the fitness (V/A)^2 vs. the chargeability (m) and the time constant (τ in sec). Both parameters show low sensitivity.

Figure 17.23 shows the field data voltage response input to the MNN and the response obtained using the inversion parameters. A two-layer model was used where the first layer was the polarizable one. There was a good match between the actual and the MNN inversion curve. The IP effect here caused the early double reversal, where the early positive was prior to the instrument early channels.

Figure 17.21 shows the fitness of this case study for the chargeability and the time constant parameters. The fitness value is close to zero at the correct values of the parameters. Large variations in this plot mean high sensitivity for the chargeability and the time constant. The fitness goes to zero at the correct values of the two parameters (m=0.7 and τ=1e-3).

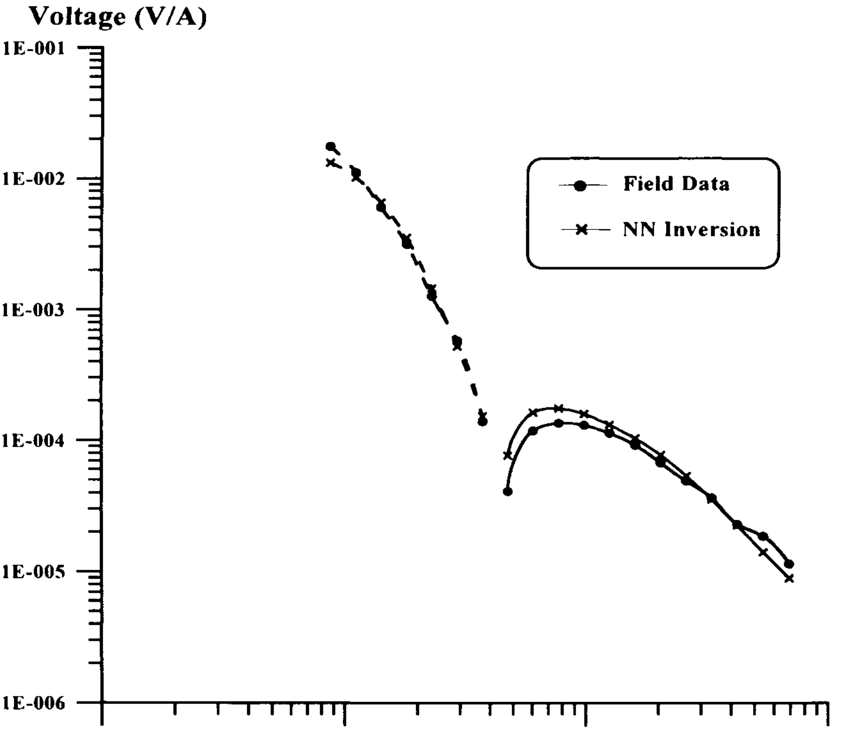


Figure 17.23. Transient voltage response from neural network inversion compared with the measured data.

Figure 17.24 shows the fitness of this case study for the first layer thickness (H) and the frequency parameter (c). The fitness value is close to zero at the correct values of the parameters. Large variations in this plot mean high sensitivity for the thickness and frequency parameter. Notice that the fitness goes to zero at the correct values of the two parameters (H=200 m and c=0.5). The second negative peak indicates a local minimum that could trap an inversion algorithm. A comparison of the network-estimated parameters and the parameters resulting in a good fitness (low error) is shown in Table 17.3.

8. CONCLUSIONS

A modular neural network algorithm was successfully used for the inversion of the TEM voltage response of a coincident loop system above layered polarizable ground to obtain its Cole-Cole parameters over a wide range of values. When the voltage response of two loops with different radii was used, the inversion was improved.

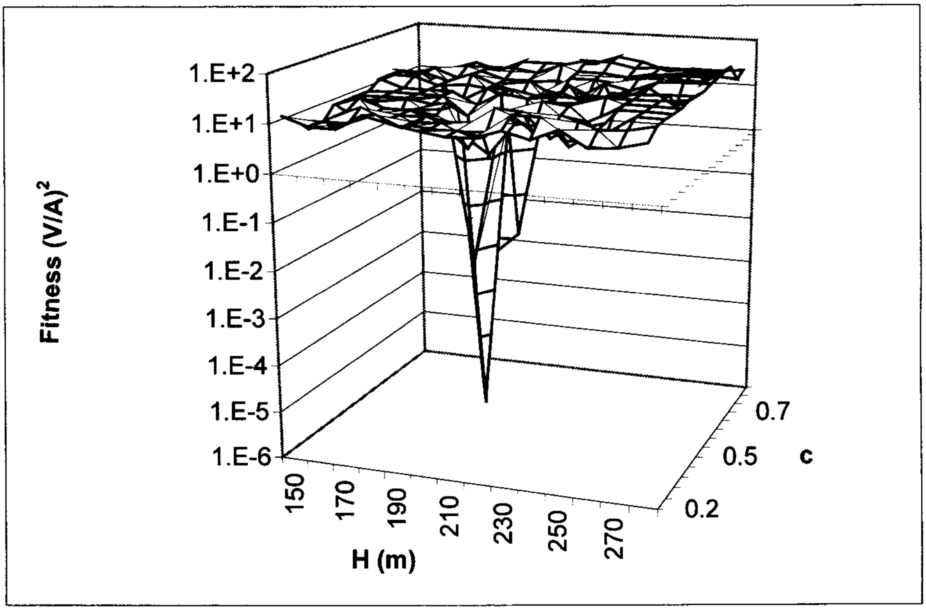


Figure 17.24. Plot for the fitness vs. the first layer thickness (H) and the frequency parameter (c).

Table 17.3
Comparison of network estimates and actual parameters for field case study

Parameters	Minimum Error Estimate	Network Estimate
m	0.7	0.65
τ	1e-3	1e-4
c	0.5	0.5
σ_1=1e-4 S/m		
σ_2=4e-3S/m		
H=200 m		

Uncertainty and sensitivity evaluations were found to be important in separating the neural network training errors from the data ambiguity problems. A confidence level was assigned to the estimated parameters by training a second network to classify the error range based on the data and previously estimated model.

The estimated parameters could be used as a starting model in any classical inversion programs to obtain more accurate results since the neural network estimates are free of trapping in far local minima.

REFERENCES

Anderson, W., 1982, A calculation of transient sounding for a coincident loop system (program TCOLOOP): USGS open-file report 82-378.

Cole, K., and Cole, R., 1941, Dispersion and absorption in dielectrics: J. Chem. Phys., **9**, 341-351.

El-Kaliouby, H., EL-Diwany, E., Hussain, S., Hashish, E. and Bayoumi, 1995, A. Effects of clayey media parameters on the negative response of a coincident loop: Geophysical Prospecting, **43**, 595-603.

El-Kaliouby, H., EL-Diwany, E., Hussain, S., Hashish, E. and Bayoumi, 1997, Optimum negative response of a coincident-loop electromagnetic system above a polarizable half-space: Geophysics, **62**, 75-79.

Flis, M., Newman, G. and Hohmann, G., 1989, Induced polarization effects in time–domain electromagnetic measurements: Geophysics, **54**, 514-523.

Johnson, I., 1984, Spectral induced polarization parameters as determined through time-domain measurements: Geophysics, **49**, 1993-2003.

Lee, T., 1981, Transient electromagnetic response of a polarizable ground: Geophysics, **46**, 1037-1041.

Oldenberg, D. 1997, Computation of Cole-Cole parameters from IP data: Geophysics, **62**, 436-448.

Olhoeft, G., 1985, Low-frequency electrical properties: Geophysics, **50**, 2492-2503.

Pelton, W., Ward, S., Hallof, P., Sill, W., and Nelson, P., 1978, Mineral discrimination and removal of inductive coupling with multifrequency IP: Geophysics, **43**, 588-609.

Poulton, M., and Birken, R., 1998, Estimating one-dimensional models from frequency-domain electromagnetic data using modular neural networks: IEEE Transaction on Geosciences and Remote Sensing, **36**, 547-555.

Smith, R., and West, G., 1988, Inductive interaction between polarizable conductors: An explanation of negative coincident-loop transient electromagnetic response: Geophysics, 1988, **53**, 677-690.

Spies, B., and Parker, P., 1984, Limitation of large loop TEM surveys in a conductive terrain: Geophysics, **49**, 902-912.

Sumner, J., 1976, Principles of Induced Polarization for Geophysical Exploration: Elsevier.

AUTHOR INDEX

A

Accrain and Desbrandes, 231
Addy, 126
Alvarez,
An and Epping, 116
Anderson, 17, 325
Anguita et al, 256
Ashida, 116

B

Baba, 87
Bak et al, 304
Baldwin et al, 231
Baum and Haussler, 65
Beard, et al, 284
Bear, 185
Benouda et. al., 231
Berg, 185
Berteussen and Ursin, 212
Beylkin, 212
Birken, 304
Birken and Poulton, 231, 304
Bishop, 52, 87, 304
Boadu, 185
Bridle, 65
Broomhead and Lowe, 304
Brown and Poulton, 231
Buffenmyer, 154
Burge and Neff, 212

C

Cacoullos, 87
Caianiello, 17, 213
Calderon-Macias, 87, 116
Calderon-Macias et al, 231
Canales, 154
Cartabia et al, 231
Cardon et al, 231
Chakravarthy et al, 284
Chen and Sidney, 116
Chen et al., 185
Chu and Mendel, 117
Cisar et al., 231
Cole and Cole, 325
Cooper, 25
Cover, 87
Cybenko, 52

D

Darken and Moody, 304
Daugman, 213
Dennis et al, 304
Dimitropoulos and Boyce, 117
Dobrin and Savit, 213
Dowla and Rogers, 185
Draper and Smith, 185

E

El-Kaliouby et al., 325
Elliott, 256
Ellis, 284
Essenreiter et al, 117

F

Fahlman, 52, 256
Fahlman and Lebiere,
Fischbach, 25
Flis et al., 325
Foster, 213
Fraser, 256
Fung et al., 231
Fu, 213
Fu et al, 213

G

Garrett, 185
Garson, 185
Geman, 52
Geman et al, 304
Gifford and Foley, 232
Girosi and Poggio, 304
Grossberg, 87
Guo et al, 232

H

Hampson and Todorov, 87, 131
Hampson et al, 126, 170
Han et al., 185, 213
Hargreaves, 140
Hargreaves et al, 154
Harlan, 213
Harrington, 256
Harrigan and Durrani, 170
Hassoun, 185
Haykin, 87, 284, 304
Hazen, 185
Hebb, 17
Hecht-Nielsen, 17, 52
Hertz, 52, 87
Hertz et al, 305
Hidalgo et al, 232
Hoff, 14
Hohmann, 256
Hopfield, 87
Hornick, 52
Hornick et al.,
Huang, 170, 232
Huang and Wanstedt,

J

Jacobs, 52, 87
James, 17
Johnson, 325

K

Klimentos and McCann, 186
Kohonen, 78
Koltermann and Gorelick, 186
Krumbein and Monk, 186

L

Lapedes and Farber, 52
Le Cun, 52
Lee, 305, 325
Leggett et al, 126
Light, 305
Lines and Treitel, 213
Link, 126
Liu and Liu, 126
Lynn et al., 154

M

Malki and Baldwin, 232
Manin and Bonnot, 154
Masch and Denny, 186
Masters, 52, 65, 98
McClelland,
McCormack, 232
McCormack et al, 117
McCulloch, 17
McCulloch and Pitts, 17, 214
Michelli, 305
Milner, 17
Minior and Smith, 232
Minkoff and Symes, 214
Minsky, 17
Moody and Darken, 305
Murat and Rudman, 117
Musavi et al, 305

N

Neff, 214
Nur et al., 214
Nyman et al., 214

O

Oldenberg, 325
Oldenziel et al, 126
Olhoeft, 325

P

Palaz and Weger, 117
Pao, 65
Papert,
Parker, 17
Palaz and Weger,
Parzen, 87
Pearson et al, 232
Pelton, et al., 325
Petros, 256
Pitts,
Poggiagliolmi and Allred, 214
Poggio and Girosi, 52, 305
Poulton, 65
Poulton and Birken, 325
Poulton and El-Fouly, 232
Poulton et al, 232
Poupon et al., 126
Powell, 305

R

Richard and Brac, 214
Rider, 284
Riedmiller and Braun, 87
Robbins, 17
Robiner and Gold, 214
Robinson, 214
Robinson and Treitel,
Rochester, 17
Rosenblatt, 17
Rosenfeld,
Roth, 117
Roth and Tarantola, 117
Roy, 284
Rumelhart,
Rumelhart and McClelland, 17, 52, 65
Rumelhart et al., 214

S

Salem et al., 232
Sacks and Symes, 214
Sen and Stoffa, 233
Sengpeil, 256
Sheriff, 214
Sheriff and Geldart, 117, 127
Shynk, 214
Skapura, 53, 87
Smith and West, 325
Solla, 53
Sontag, 256
Sommen et al., 214
Spath, 305
Specht, 88
Spichak and Popova, 233
Spies and Parker, 325
Sternberg and Poulton, 306
Sumner, 325
Swingler, 53, 98
Swiniarski, 233

T

Tarantola and Valette, 215
Taylor and Vasco, 233
Treitel et al., 215
Thomas, 306
Tikhonov and Arsenin, 306
Todorov et al., 127

V

Veezhinathan and Wagner,
Veezhinathan et al, 118
Vernik, 215
Vernik and Nur, 224
von Neumann, 17, 25

W

Walls et al., 127
Wait, 256
Wang and Mendel, 88, 118
Wasserman, 88
Werbos, 18, 53
Whitman, 284
Whitman et al, 284
Widrow, 215
Widrow and Hof, 18, 53
Widrow and Stearns,
Wiener et al., 233
Wittner and Denker, 53
Wong et al., 233

Y

Yang and Ward, 285
Yilmaz, 118, 154
Yoshioka et al., 127

Z

Zell, 306
Zhang et al, 284
Ziolkowski, 215

This Page Intentionally Left Blank

INDEX

A

acoustic impedance, 193
activation, 20, 30, 194, 311
ADALINE, 13, 14,
Adaptive Resonance Theory, 84, 96, 97, 109
airborne electromagnetic measurements, 235, 236, 252
  profiles, 247, 248
  responses, 250
  survey, 219
  system, 236
ambiguity, 311, 312, 315, 320, 321, 325
apparent resistivity, 257
applications, 3, 5, 16, 77
architectures, 16, 23, 155, 224, 294
  ART1, 84
  Boltzmann Machine, 224
  Caianiello network, 195
  Functional link network, 64
  Hopfield, 81, 82, 83, 84, 96, 113, 114, 115
  LVQ, 222
  MLP, 67, 119, 266
  MLP alternatives, 67
  recurrent, 50
  RBF, 266
  RProp, 267
ART1 *See* Adaptive Resonance Theory
associative memory, 6, 24, 81, 82
asymmetrical transfer functions, 241
automatic inversion, 261
AVO, 123, 126, 134, 153,
axon, 10

B

back-propagation, 11, 16, 24, 27, 34, 39, 60, 64, 91, 178
  examples, 110
basement highs, 224
batch learning, 34
Bayesian, 192, 193
bias, 14, 28, 30, 35, 40
  weights, 39, 50
binary output coding, 60
binary variables, 60
biological neural networks, 3, 9, 19, 23
Blackfoot field, 123, 155, 157
block-component method, 113
Boltzmann Machine architecture, 224
boot-strap. *See* cross-validation
boreholes, 257, 258, 260
Born approximation, 189
brain, 3

C

Caianiello neural network, 187, 193, 195, 201, 206, 211
Caianiello neuron, 14, 187, 188, 194, 195, 210
  equation, 187, 188, 194
cascade-correlation algorithms, 107, 108
cerebral cortex, 23
classification, 103, 107, 135, 206
  by SOM, 166
  of seismic responses, 155
  of seismic traces, 161
  supervised, 119
clastics, 222
clay content, 174, 176, 177, 181, 184, 206, 213, 221
cluster analysis, 236
CMP stacking, 131, 145
coal mine, 287, 297, 303
cokriging, 124
Cole-Cole model, 307, 313, 316, 320, 324
commercial packages, 89, 92
  Brainmaker, 93
  EMERGE, 123
  EMIGMA, 239
  Geovecteur, 150
  MATLAB, 73, 181, 266
  NeuralWare, 89, 267, 274
  NeuroShell, 92, 93, 94
  NeuroSolutions, 90, 92
  SPSS, 93, 94
  STASTICA, 94, 95
common midpoint, 110, 111, 150
competition. *See* winner take all
competitive learning, 158, 165
complex conductivity, 310
compressional velocity, 174
computation time, 299
computational PEs, 21
concentration gradient, 20
confidence level, 140, 147, 148, 149
conjugate gradient, 71, 72, 74
connection strategy, 27
connection weights, 8, 13, 14, 19, 22, 27, 30, 31, 34, 39, 40, 41, 42, 46, 48, 50, 194, 263, 267
  hidden layer, 264, 265, 267
  for Hopfield, 81
  intializaton of MNN, 264
  for Kohonen, 80
  MNN, 263
  RBF, 293
  RProp, 267
context units, 51
continuous variables, 60
continuous-valued mapping, 63
convergence, 35, 44
  rate, 178, 197
convolution problem, 113
convolutional neuron model. *See* Caianiello neural network
cooperatition, 27
coring, 220
cost function, 196, 198
  entropic, 48
CREWES project, 157
cross-validation, 123, 126

D

data distribution, 58
data transformation, 57
DC conductivity, 307, 310
DC resistivity, 307
deconvolution, 101, 112, 114, 117, 131, 150, 190, 200, 204
Debye model, 310
delta bar delta, 46, 67
  modified, 68
delta learning rule, 14, 104, 127, 134
dendrite, 21
density logs, 260
Denver-Julesberg Basin, 224, 232
detection of 2D structures, 236, 244, 250
dielectric permittivity, 238
dikes, 236, 250, 252
  detecting, 252
  structures, 235, 248, 250
dip, 129, 132
directed random search, 68
  algorithm, 69

E

earth model parameter estimation
  1D, 226
eddy currents, 307
electrical resistivity, 219
elimination of multiples, 112
electrical conductance modeling, 235
electromagnetics, 225
  ellipticity, 226, 231, 287
  frequency-domain, 232, 287
  magnetotelluric, 227
  methods, 307, 309
  properties, 307
  surveys, 219
ellipticity, 58, 61, 73, 287, 288, 289, 294, 298
  3D, 287, 289
Ellipticity Data Interpretation and visualization System, 288
EM fields
  calculation of, 239
entropic cost function, 49
epoch, 34
equivalence, 311
error, 288, 293, 303
  accumulation, 35
  bias, 40, 289
  correction, 31
  funtion in LM, 171, 179
  synthetic, 249
  training, 297, 312, 317
  variance, 40, 289
environmental surveys, 229

F

facies, 119, 120, 260
  mapping, 155
facies analysis log, 222
false alarm, 270, 275, 278, 281, 284
false boundaries, 269
FAR peak, 110
feature extraction, 61
feed-back, 9
feed-forward network, 20, 102, 178, 290
finite differences method, 239
finite elements method, 239
fitness, 321, 322, 323, 324
Fogo volcano, 321
forward modeling, 206, 219
  1D, 237, 247, 252
  2D, 239
  3D, 239
first break picks, 101, 103, 104, 105, 109
Fredholm integral equation, 190
frequency domain, 287, 309, 310
frequency parameter, 323, 324
frequency response, 310
fully-connected, 20
function approximator, 171, 289
functional link network, 64
function mapping, 24

G

Gabor function, 195, 196
Gaussian basis functions, 172, 174
Gaussian distribution, 44, 173
Gaussian mixture model, 264
Gaussian noise, 265, 268, 269
Gaussian transfer function, 268, 278, 283, 284
general regression neural network, 266, 269, 274, 280, 283
  in MATLAB, 266
  spread varible, 268, 284
generalization, 289, 290
genetic approach, 221
GRNN *see* general regression neural network
gradient descent, 35, 48, 242
geophysical instruments
  EM38, 230
  EM61, 227
  GRS-1, 230
  PROTEM EM47, 321
geophysics, 3, 4, 5, 65
  parameters, 172
geophysical gradient descent, 178
geophysical inverse problems. *See* inversion
geophysical surveys, 297
Goupillaud earth model, 199
gradient of the error, 228
grain size, 309
gravity, 219, 224
  gradiometry, 225, 233
  surveys, 219
ground penetrating radar, 229
groundwater, 235

H

half-space, 287, 295, 299, 311
  neural network, 294
  piecewise, 298
  piecewise networks, 288, 294
Hankel transform, 310

- Hebbian learning, 9, 11, 24
- Hessian matrix, 42, 71
- heteroassociative, 27
- hidden layers, 34, 36, 38, 241, 249
  - building, 41
  - MLP, 223
  - number of, 35
  - RBF, 90, 266
- hidden PEs, 19
  - number of, 35
- High Definition Induction Log, 262
- homogeneous half-spaces, 235, 239, 243, 244, 246, 248
  - parameters, 244
- Hopfield network, 81, 82, 83, 95, 96, 113, 114, 115
- horizons, 148
  - picking, 119, 155
  - tracking, 119, 120, 155, 157, 161
- hydraulic conductivity, 176, 177
- hybrid network, 120
- *hybrid* RBF, 293
- hyperbolic reflection signatures, 229
- hyperbolic tangent, 31
- hypersurface reconstruction, 289

I

- ill posedness, 190
- impedance logs, 190, 202, 205, 209
- impedance model, 212
- induced polarization, 256, 325
- information theory, 187
- input layer, 11
- input signal reconstruction, 187, 188, 198
- interconnected strategy, 27
- interference
  - low-frequency, 137
- invasion zone, 260, 262
- inverse modeling, 219, 310
- inverse transform, 191
- inversion, 174, 187-195, 198-201, 203, 206-212, 220, 227, 230, 233, 242, 256, 260, 261, 284, 288, 298, 301, 303, 304, 309, 311, 312, 315, 323, 324, 325
  - improving, 311
- integral equations method, 239
- IP. *See* induced polarization
- iteration, 31
- iterative inversion, 242

J

- Jacobian matrix, 228
- joint impedance, 200
- joint inversion, 190, 199, 200-204, 207, 210

K

- kernel function, 188, 189, 191
- Kirchhoff modeling, 113
- K-means clustering, 165, 223
- Kohonen, 78
- Kohonen layer, 79, 80
  - architecture, 97
  - two dimensional, 155, 157
- Kohonen network, 16, 94
- Kolmogorov Theorem, 35, 36
- Kozeny-Carman model, 177
- kriging, 124, 192

L

- landslides, 235
- late time voltage response, 310
- layer thickness, 219
- layer boundaries, 257, 258-262, 268
  - locating, 257
  - picking, 274, 278
- layer models
  - medium, 273, 275
  - thick, 277, 278
  - thin, 268
- layered-earth models, 288, 299, 303
  - lD layered parameters, 236
- learning algorithm, 15, 16, 241, 242
- learning rate, 13, 35, 45, 48, 52, 90
- learning rule, 34
- learning strategy, 27
- least-squares polynomial fitting, 175
- least-mean squares, 14, 27, 219, 228
  - inversion, 297, 305
- Levenberg-Marquardt, 72, 171, 178, 261
- lineament, 224
- Lipmann-Schwinger equation, 189
- lithofacies, 221, 222
  - identification, 220, 222
  - mapping, 220, 221
- lithologic
  - mapping, 224
- lithology, 206, 222, 231,
  - carbonate, 193, 194
  - conglomerate, 223
  - joints, 193
  - mapping, 220
  - sandstone, 206, 207
  - shale, 206, 207
- local experts, 262, 266, 271, 272, 310
- local minimum, 43
- logistic function, 35, 56
- logging tools
  - electrical current, 257
- high definition induction, 262
  - galvanic, 220
  - gamma ray, 222, 224
  - geophysical, 224, 260
  - laterolog deep, 220
  - multiple, 262
  - neutron, 220, 224
  - resistivity, 224, 257
  - spherically focused logs, 260
  - unfocused, 257
- Lyapunov energy function, 83

M

- magnetics, 224
- mapping function, 290, 291, 295
- McCulloch-Pitts neuron, 7-11, 15, 187, 188, 194, 195
- mean field annealing, 114
- mean squared error, 31, 321

- memory, 22
- MLP *See* multi-layer Perceptron
- MNN *See* modular neural network
- modular neural network, 75, 226, 232, 262, 265, 310, 324
- momentum, 45, 46, 241
- monotonic functions, 35
- MS wavelet, 202, 203, 204, 205, 209
- MSI wavelets, 202, 203, 204, 205
- mud resistivity, 260
- multi-layer Perceptron, 20, 27, 28, 30, 240, 291
  - gradient descent learning, 267

N

- nearest neighbor, 293
- neighborhood
  - in Kohonen layer, 156
- network architecture, 134, 144, 189, 190, 195
- network design, 56, 180
- neural wavelet, 194, 195, 203, 207-210, 212
  - estimation, 187, 188, 196, 201, 203, 210
- neuron, 14, 19, 20, 21, 24
- neurotransmitters, 21, 22, 23
- Newton-Raphson technique, 260
- news groups, 97
- NMO correction, 110, 111, 112
- non-genetic approach, 221
- noise, 35, 40
  - in seismic traces, 108
- nonlinear factor optimization, 187, 198
- nonlinear relationships, 172
- nonlinear transform, 189, 193, 201, 207
- non-polarizable layer, 316
- normal moveout velocities, 131, 134
- novel signature, 220
- novelty detection, 220

O

- Oklahoma Benchmark earth model, 261, 262
- open source software, 97
- output
  - binary valued, 84
- output layer, 19, 27, 28, 32, 34, 40
- output PE, 28, 32, 39, 40, 46

P

- pattern mode, 48
- pattern recognition, 52, 61, 63, 134
- pattern unit, 267, 269, 278
- PE. *See* processing element
- performance, 34, 38, 40, 47, 124, 129
  - in SOM, 165
- Perceptron, 11, 12, 13, 24, 291
  - Mark I, 11
- permeability, 171, 172, 174, 176, 178, 184, 220
  - estimation, 171
  - in carbonate, 220
  - magnetic, 238
- petrophysical properties, 172, 173
- piecewise half-space, 288, 294, 295, 296, 298. *See* half-space neural network, 298, 302, 303, 304
- piecewise half-space interpretation neural networks, 294
- piecewise inversion, 299, 300, 302, 303
- plasticity, 84
- PNN *See* Probabilistic Neural Network
- Poisson's Ratio, 123
- pore pressure, 221
- porosity, 119-126, 155, 161-164, 167-169, 172, 174-177, 180, 181, 184, 185, 188, 193, 206-208, 211, 220-223
  - seismic estimation, 162
- Poulter source, 106
- pre-frontal cortex, 22
- pre-processing, 56
  - example, 109, 134
  - re-scaling, 56
  - with SOM, 120
- principal components analysis, 223
- probabilistic neural network, 75
- processing element, 19
- profile segments, 236, 243, 250, 251
- pruning, 27, 41, 42

Q

- quadratic cost function, 48, 49
- quasi-Newton method, 72, 179, 180
- QuickProp algorithm, 228

R

- radial basis functions, 49, 74, 75, 172, 174, 184, 227, 262, 267, 287, 289, 290, 292, 296
  - architecture, 294
  - *hybrid*, 293
  - training, 173
- radial-distribution functions, 172
- RBF *see* radial basis functions
- random models, 248, 249, 250
- recognition system
  - for steel drums, 225
- recurrent networks, 50
- regularization, 35, 49
  - networks, 290
- reservoir characterization, 119, 121
- Resilient back-propagation. *See* RProp
- resistive beds, 258
- resistivity, 222, 228, 229, 245, 246, 249, 257-261, 265, 268, 272-274, 277, 280, 283, 288, 289, 298-301, 303
  - sections, 298, 299
  - sensitivity to, 280
- resistivity-depth sections, 298, 299, 300
- ridge regression procedure, 260, 261
- Riemann-Lebesgue theorem, 190
- rms error, 37, 39, 46, 228, 297, 303, 311-313, 315-320
- Robinson seismic model, 188, 189, 199
- RProp, 267, 269, 270, 274, 275, 278, 279, 280
  - architecture, 267
  - algorithm, 70

S

- scaling, 56, 57, 60
- scatter distribution, 193, 207, 208, 211
- seismic attribute, 101, 105, 122-125
- seismic bandwidth, 199
- seismic convolutional model, 188, 190, 199, 200, 201, 211

seismic crew noise, 129, 131
  front-end interference, 132, 133
  rear-end interference, 131, 133
  sideswipe interference, 132, 133
seismic parameters, 172
seismic section, 203, 205
seismic signatures, 119
seismic survey
  3C-3D, 123
  4D, 121
seismic trace, 101, 104, 106, 110, 113, 155, 161, 200-204
seismic wavelet, 199, 200, 203, 204, 211
  estimation, 199, 200
seismograms, 199
self-organizing map, 16, 78, 79, 120, 155, 226
  as filter, 120
  architecture, 155
  number of classes, 158, 160, 165, 166, 168
sensitivity analysis, 321
Sigma-Pi units, 78
sigmoid function, 22, 30, 43, 46, 56, 267
sigmoidal threshold, 23
signal processing, 187, 188, 195, 210
simplified model, 199
singularity, 190, 191, 192
size reduction, 56, 58
splines, 289, 290
SNARC, 10, 11, 13
softmax function, 60, 264
software. *See* commerical packages
SOM *See* self organizing map
soma, 19
sonic log data, 122
spatial aliasing, 137
spatial relationships, 61
spontaneous potential, 222
standard regularization, 289
step size, 36, 45, 46, 293
stratigraphy, 297
subsidence, 287, 297, 298, 299
sum-of-squares error function, 179
supervised learning, 19, 27, 119, 120, 262, 293
  for Kohonen, 81
supervised network, 221, 222
synapse, 21, 22
synthetic logs, 260
synthetic seismograms, 121, 123, 124

T

tanh, 35, 43, 49, 56, 60, 266, 267
target parameters, 220, 226
TEM. *See* time domain
terrain mapping, 235
testing, 34, 40, 48
threshold, 8, 10, 12, 13, 14, 194, 224, 243, 251
threshold function, 30, 31, 43, 49
time constant, 307, 308, 310, 321, 322, 323
time-domain, 197, 220, 227
time signal, 188, 194, 195
time-varying data, 50
topology, 156, 157
trace editing, 101, 109
training, 7, 11, 27-30, 34, 38-40, 42, 44, 48, 55, 57, 64
  of MLP, 35
  MNN, 263, 266, 271, 283
  RBF, 171, 173, 178, 181
training set design, 129, 134, 145
transfer functions
  cubic sigmoid, 241, 245, 249
  hyperbolic arctangent, 242
  hyperbolic tangent, 31, 241
  hypsigmoid_241, 247, 249
  sigmoid, 240, 242
transient voltage response, 308, 323
two-layer earth model, 310, 312

U

uncertainty evaluation, 320
unexploded ordnance, 219, 227
uniqueness, 187, 211

V

validation, 34, 40, 48
velocity analysis, 101, 110, 115
very fast simulated annealing, 224, 229
Vibroseis data, 106, 107, 108
vision, 195
visualization shell, 288
voltage pattern, 320
voltage ratios, 243, 244, 249, 251, 252

W

wave propagation, 189, 199
waveform recognition, 101
weight changes, 32, 33, 45
weight initialization, 35, 43
weight update
  in Hopfield network, 115
  in LM, 180
winner take all, 11, 27
well logging, 155, 161, 177, 193, 203, 204, 206, 219, 220, 221, 223, 260
window length, 165, 166
  in seismic data, 157

This Page Intentionally Left Blank